File No. S360-30
GY28-2016-5

IBM

Program Logic

Version 8.1

IBM System/360 Time Sharing System

Access Methods

This publication describes the internal logic of the nonresident access methods used in TSS/360. (The facility which controls most conversational input/output in TSS/360, the resident terminal access method (RTAM) is part of the resident supervisor and is described in IBM System/360 Time Sharing System: Resident Supervisor Program Logic Manual, GY28-2012.)

The nonresident access methods are:

- The virtual access methods (VAM), used to store and retrieve page-organized data located on direct access devices, and including the virtual sequential (VSAM), virtual index sequential (VISAM), and virtual partitioned (VPAM) access methods.
- The sequential access methods (SAM), used to access OS/360-oriented data sets on tape or direct access devices, and including the basic sequential (BSAM) and queued sequential (QSAM) access methods.
- The multiple sequential access method (MSAM), used for efficient input/output with unit record equipment.
- A facility, IOREQ, which allows a user to provide his own access method with a private device.
- The terminal access method (TAM), which allows input/output with specific terminals.
- A Terminal Task Control module which provides task control for multiterminal task (MTT) applications.

For each access method, an overview, routine descriptions, and flowcharts are provided.

This material is intended for persons involved in program maintenance, and system programmers who are altering the program design. It can be used to locate specific areas of the program, and it enables the reader to relate these areas to the corresponding program listings. Program logic information is not necessary for the use and operation of the program.

PREFACE

This publication describes the access methods in TSS/360 (except for the resident terminal access method*). It can be read selectively for a general understanding of a particular access method, or it can be used as a guide to more detailed information in an object program listing of a particular access method object module.

HOW THIS BOOK IS ORGANIZED

The access methods are grouped in this book into four parts:

- Basic Sequential Access Method (BSAM), Multiple Sequential Access Method (MSAM), Terminal Access Method (TAM), and IOREQ.
- Virtual Access Methods (VAM), including Virtual Sequential (VSAM), Virtual Index Sequential (VISAM), and Virtual Partitioned (VPAM) Access Methods.
- Queued Sequential Access Method (QSAM).
- Terminal Task Control (a facility which controls tasks that have Multi-Terminal Task (MTT) applications).

For each access method, an overview and individual routine descriptions are provided. Flowcharts for all access methods are grouped in one section.

TO USE THIS BOOK, YOU NEED:

General familiarity with TSS/360 assembler language and the main concepts of TSS/360.

GENERAL BOOKS TO REFER TO:

IBM System/360 Principles of Operation, GA22-6821.
IBM System/360 Time Sharing System: Concepts and Facilities, GC28-2003.

BOOKS RELATED TO ACCESS METHODS:

Access methods are usually invoked as the result of user- or system-initiated macro instructions described in:

IBM System/360 Time Sharing System: Assembler User Macro Instructions, GC28-2004.

A complete explanation of data management in TSS/360 from the user point of view is provided in:

IBM System/360 Time Sharing System: Data Management Facilities, GC28-2056.

TO FIND AN ACCESS METHOD:

See the Table of Contents.

TO FIND A PARTICULAR ROUTINE (MODULE):

See Appendix C, Access Methods Module Directory.

---

*See Resident Supervisor Program Logic Manual, GY28-2012.

Fifth Edition (September 1971)

This is a minor revision of GY28-2016-4 incorporating TNL GN28-3212.

This edition is current with Version 8, Modification 1, of the IBM System/360 Time Sharing System (TSS/360), and remains in effect for all subsequent versions or modifications of TSS/360 unless otherwise noted. Significant changes or additions to this publication will be provided in new editions or Technical Newsletters. Before using this publication, refer to the latest edition of IBM System/360 Time Sharing System: Addendum, GC28-2043, which may contain information pertinent to the topics covered in this edition. The Addendum also lists the editions of all TSS/360 publications that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Department 643, Neighborhood Road, Kingston, N.Y. 12401

TABLES

PART I

ACCESS METHOD FOR BSAM, MSAM, TAM AND IOREQ

The data management access methods for IBM Time Sharing System/360 (TSS/360) include the routines, control blocks, and work areas that receive or transmit data from or to I/O devices. Part I describes data management access methods:

1. Basic Sequential Access Method (BSAM)
2. Multiple Sequential Access Method (MSAM)
3. Resident Terminal Access Method (RTAM) - Overview only
4. Terminal Access Method (TAM)
5. I/O Request (IOREQ)

The routines in each of the access methods, although similar in operation, differ in these ways:

- BSAM routines enable the user to access data at the READ/WRITE macro instruction level. BSAM processes sequential data sets that reside on magnetic-tape or direct access devices. BSAM data sets are compatible (except for the two limitations indicated in the BSAM overview) whether created under IBM System/360 Operating System, referred to as Operating System/360 (OS/360), or under TSS/360. Both data sets can be processed by OS/360 BSAM and TSS/360 BSAM.

- MSAM routines allow the user to process logical records at the GET/PUT macro instruction level for the 2540 card reader/punch and 1403 printer. MSAM differs from BSAM in that the channel command words (CCWs) used to perform the I/O operations on the above mentioned devices are command chained, significantly reducing the interruption processing overhead. MSAM can be employed by any user; however, device management restricts the use of unit record equipment to privileged users.

- RTAM routines, as the name suggests, are located mainly in the resident supervisor. For the logic flow and detailed descriptions of the major portion of RTAM, see the System Logic Summary PLM, GC28-2009 and the Resident Supervisor PLM, GY28-2012. This publication describes Terminal Task Control, the virtual storage routine, analogous to other access methods routines, which initiates multi-terminal tasks (MTT) and sets up read, write, and polling control blocks for READQ, WRITEQ, FINDQ, FREEQ, and CLEARQ macros issued by the MTT user.

- TAM routines restrict nonprivileged user programs to accessing data using the GATE macro instructions, and allow privileged programs to access data at the READ/WRITE macro instruction level. TAM processes sequential data sets that reside on specified communication terminals.

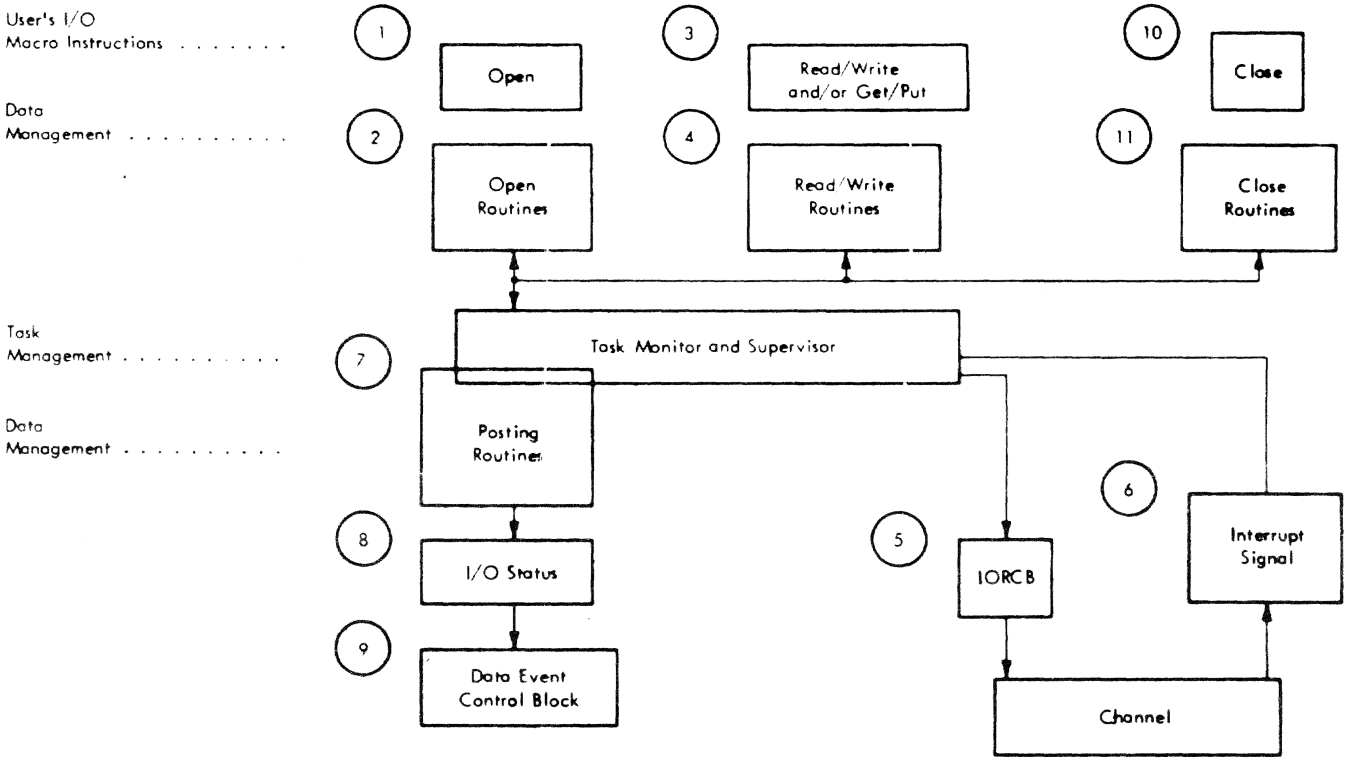
- IOREQ routines permit the user to access data, from any device, at the channel command word (CCW) level. The user must, however, be privileged to have access to these IOREQ routines for unit record equipment.

This section provides an introduction to the access methods, the user's I/O macro instructions, the common control blocks, and contains an overview of BSAM, MSAM, TAM, and IOREQ. Sections 2, 3, 4, and 5 discuss details of the access method routines' Open, Read/Write, Posting and Close phases. Section 6 discusses routines primarily designed for use with BSAM.

Linkage to the Access Methods Routines

Problem programs, as well as IBM-written system programs, may, directly or indirectly, use the four access methods (BSAM, MSAM, TAM, and IOREQ), therefore privileged and nonprivileged programs may link to these access methods. The access routines are called upon by I/O macro instructions that are in source programs. During language processing, the I/O macro instructions are expanded into code that links to and passes parameters to appropriate access method routines. The DCB, DFTRMENT, and DCBD macro instructions do not link to access methods routines but complete their function during assembly. Expansion of the DCB and DFTRMENT macro instructions only build control block tables. The DCBD macro instruction inserts a dummy control section at the place the macro instruction is encountered.

Privileged programs use Type-1 linkages to the privileged access method routines; nonprivileged programs use the ENTER mechanism. See IBM System/360 Time Sharing System: Task Monitor Program Logic Manual, GY-28-2041, for an explanation of ENTER.



Note: Numbered boxes refer to the accompanying description under Access Method Phases.

Figure 1. Access Method Phases for BSAM, MSAM, TAM, and IOREQ

Access Method Phases

Each of the BSAM, MSAM, TAM or IOREQ access methods may be presented in four phases, illustrated in Figure 1.

Here are some preliminary notes on each of the four phases. The numbers in parentheses correspond to those circled in Figure 1.

Open Phase: Details are presented in Section 2 of this part.

(1) If a user desires transmission of data to or from an I/O device, he must initially use the OPEN macro instruction in his program.

(2) The OPEN macro instruction calls the open routines that prepare the I/O devices, control blocks, and the data set for further processing.

Read/Write Phase: Details are presented in Section 3.

(3) After completion of the OPEN routines, data may be transmitted by using the READ/WRITE macro instructions.

(4) These call the READ/WRITE routines that build CCWs in the IORCB.

(5) The resident supervisor is requested to execute the channel commands via an IOCAL SVC.

Note: The read/write phase is initiated by READ/WRITE or GET/PUT level macro instructions where the specific access method READ/WRITE or GET/PUT level macro instructions are listed in Table 1. READ or GET macro instructions read data from an I/O device. WRITE or PUT macro instructions either write data to an I/O device or send control information to the control unit.

Posting Phase: Details are presented in Section 4.

(6) After an I/O operation has been completed, an I/O interruption occurs which results in the storage of I/O status information and the IORCB at fixed locations in the interruption storage area (ISA).

(7) The posting routines obtain control, via the task monitor, after the resident supervisor receives the interruption, with all other task interruptions masked off.

Table 1. BSAM, MSAM, TAM and IOREQ READ/WRITE and GET/PUT Level Macro Instructions

Access Method	READ Macro Instruction	WRITE Macro Instruction
BSAM	READ	WRITE
MSAM	GET	PUT
TAM (privileged)	READ	WRITE
TAM (nonprivileged)	GATRD	GATWR
IOREQ	VCCW, IOREQ	VCCW, IOREQ

(8) These posting routines record whetner the I/O operation had a normal or abnormal completion.

(9) The posting routines post the results in a DEB and a DECB.

Note: Error retry and recovery is attempted by the system, if the I/O completion was abnormal. (IOREQ POSTING does not have error retry or recovery routines.) The posting routines return control to the task monitor which returns control to the interrupted program by loading the old I/O VPSW. Consequently, the posting processing is transparent to data management routines except for the altered DEB and DECB. The CHECK macro instruction in the user's program waits for and checks completion of the I/O operation that is posted in the DECB. (MSAM does not have a CHECK macro instruction. DOMSAM performs the functions of the CHECK macro for MSAM. GET/PUT provides return codes which indicate the completion status of the requested operation.)

Close Phase: Details are presented in Section 5.

(10) When the user has completed processing his data set he issues a CLOSE macro instruction.

Note: When the MSAM user has completed processing his data group he may issue a FINISH macro instruction prior to the CLOSE macro instruction to empty or truncate the last buffer and test the result of all outstanding I/O. This is preferable to simply issuing CLOSE since the task is permitted to continue as the I/O operation queue.

(11) CLOSE calls the close routines which will reset/release control blocks and complete data set processing.

Macro Instructions

To communicate with the access methods, specific I/O macro instructions in the user's program are required for BSAM, MSAM, TAM and IOREQ (see Table 2).

All I/O macro instructions available for each of the access methods are in Table 3. For more details, see Assembler User Macro Instructions.

Control Blocks

The list below indicates the control blocks that are within the user's program and access method storage; BSAM, MSAM, TAM, and IOREQ require that all these control blocks be generated if the access methods are to function. Additional control blocks are described later, under the access methods to which they are related.

Assembly Generated Control Blocks	Dynamically Built or SYSGEN Generated System Control Blocks
	SDAT
	SDT
DCB	JFCB
*DECB	DEB
	IORCB

*In MSAM the DECB is generated at open time.

Control blocks are generated either in the user's program area or in the access method area.

USER'S PROGRAM AREA: The following control blocks are generated in the user's program area.

Data Control Block (DCB): The DCB is generated during assembly by a DCB macro instruction and serves as a basic communications area for I/O operations. It is used to maintain information such as data set organization, attributes of data set buffering information (used in data set processing), and addresses of special exit routines.

Data Event Control Block (DECB): The DECB is generated during assembly by a READ/WRITE macro instruction (except for MSAM) and serves as an I/O status-reference block for I/O operations. It is used to store information such as: state of completion of an I/O operation, type of operation previously issued, CSW information, and codes that indicate, to the user program, conditions on ending the I/O operation.

ACCESS METHOD AREA: The following tables and control blocks are generated in the access methods area.

Table 2. BSAM, MSAM, TAM, and IOREQ I/O Macro Instructions Required for I/O Operations

I/O Macro Instruction	Explanation
DCB	Required prior to open phase; reserves space in user's program for the data control block (DCB), which is basic communication area for I/O operations.
OPEN	Required to start open phase, links to open routines to open the DCB.
GET/PUT	Required to initiate transmission of data. Applicable only to MSAM.
READ/WRITE*	Required to initiate transmission of data; reserves space for the data event control block (DECB), which contains status of I/O operations; links to read/write routines that build CCWs in the IORCB, and causes the commands to be executed by issuing the IOCAL SVC.
CHECK	Required under BSAM, TAM, and IOREQ to test I/O operation associated with the DECB; waits for and checks completion of I/O operations that are posted in the DECB; for I/O normal completion returns to problem program; for exception I/O completion exits to the routines specified in the DCB.
CLOSE	Required to disconnect data set from user's program; links to close routines that complete control blocks.
DCBD	Required if user accesses the DCB fields; it provides the dummy control section (DSECT) which contains all symbolic names used to reference information in a DCB.

*Refer to Table 1 for listing of specific READ/WRITE macro instructions. READ/WRITE macro instructions at assembly time either generate space for a DECB (S and L form) or refer to an existing DECB (E form); MSAM has no READ/WRITE macro.

Table 3. BSAM, MSAM, TAM, and IOREQ Macro Instructions

Used in Common	BSAM	MSAM	TAM	IOREQ
DCB	READ	GET	READ*	IOREQ
DCBD	WRITE CHECK	PUT SETUR	WRITE* CHECK	VCCW CHECK
OPEN	GETPOOL	FINISH	DFTRMENT	
CLOSE	GETBUF FREEBUF FREEPOOL NOTE POINT BSP FEOV CNTRL CLOSE (TYPE=T) DQDECB			

*In TAM, only privileged programs may issue the READ/WRITE macro instructions. Nonprivileged programs issue GATRD/GATWR macro instructions.

Symbolic Device Allocation Table (SDAT): The SDAT which is initialized by device management resides in public virtual storage and provides information on the status and characteristics of each allocatable I/O device in the system. The SDAT contains information on the symbolic device address, model code, device code, device class and unit type.

Job File Control Block (JFCB): The JFCB is constructed for a data set by the DDEF routine from information in a DDEF command or macro instruction. The information in the JFCB is used to complete the DCB during execution of the OPEN macro instruction. The JFCB contains information defining the data set attributes, information on where the data set is located, and pointers to other JFCBs in the task.

Note: The data set organization must be specified in the DDEF command for IOREQ.

Data Extent Block (DEB): The DEB is constructed at OPEN time for a given data set and serves as a data set reference block for I/O operations. The DEB is used to store information such as volume locations, device and data set attributes, pointers to other control blocks associated with the data set, and pointers to DECBs that have not been checked.

I/O Request Control Block (IORCB): The IORCB is constructed at READ/WRITE time and serves as a control reference block for I/O operations. It contains the CCWs that control the I/O operation and may contain either the I/O data buffer, or a pointer to the I/O data buffer.

I/O Statistical Data Table (SDT): For each device the SDT contains maximum retry thresholds which are used in certain error processing.

BSAM OVERVIEW

BSAM comprises those instructions, control blocks and data areas which allow for limited data set interchange between TSS/360 and OS/360. (TSS/360 BSAM will not support the OS/360 direct access split cylinder format and will not deblock records provided by OS/360 QSAM.)

With tape input and output, BSAM supports either EBCDIC symbols and the standard IBM label and record formats or the print symbols and label and record formats of the American National Standard Institute. The latter print symbols standard is officially the American National Standard Code for Information Interchange X3.4-1968, and is referred to herein as ASCII. The latter label and record formats standard is officially the American National Standard Magnetic Tape Labels for Information Interchange, X3.27-1969, and the format is referred to herein as American National Standard. TSS/360 processing is in EBCDIC and standard IBM format; when the user specifies the ASCII option as a parameter of the DDEF command, BSAM provides a conversion interface between ASCII and EBCDIC symbols and American National Standard and standard IBM formats.

The access methods descriptions presented in this manual do not include the DDEF command, which performs some preliminary open functions for BSAM. The Open Common and Open BSAM routines oversee the completion of the necessary open functions.

The DDEF command contains such information about the data set as its name, volume residence, organization, and type of device used. The DDEF command causes the building of the job file control block (JFCB), oversees initial device allocation and mounting and allocates space for data sets on direct access storage devices.

By macro instruction, the user will be linked to the Open Common routine. This routine performs those open functions which are common to all the TSS/360 access methods. These functions basically are:

- Finding the JFCB in the system.
- Filling in the data control block (DCB) with information from the JFCB. The user is thus able to specify, for a particular run, many data organization and handling options which may not be known at assembly time.
- Ensuring that only privileged programs use privileged data sets.
- Checking for conflicts between user indicated options and control block data.
- Getting a page of storage to be used later for input/output request control blocks (IORCBs).

For BSAM, Open Common links to BSAM Open. At this point, the JFCB and DCB have been constructed, a data extent block (DEB), the primary control block used by the Read/Write routines for such information as the device type, error statistics, outstanding IORCBs, and the queue of unchecked I/O requests, is still required. BSAM Open's main function is the building of this DEB in protected storage so that it cannot be destroyed or changed by the user program. To build the DEB, linkage is made to the Build Common DEB routine or to the Build DA DEB routine; the choice depends on device type.

DA Open is used to complete open processing for direct access devices while Tape Open is used to complete the open process for tape. Both DA Open and Tape Open call the appropriate label processing routines to process user and data set labels. Label information such as record length can be used to modify the DCB.

The labeling routines available in BSAM are Tape Label Processor, which has separate entry points for input header, output header, input trailer, and output trailer labels, DA Input User Label Processor, and DA Output User Label Processor.

Where ASCII-encoded tapes have been specified, the ASCII Translation and Conversion routine provides ASCII-to-EBCDIC translation on tape input and EBCDIC-to-ASCII translation on tape output.

Four macro instructions are provided to obtain buffer space: GETPOOL, GETBUFF, FREEPOOL, and FREEBUFF.

After his data sets have been opened, the user will normally access data by means of a READ or WRITE macro instruction, followed (not necessarily immediately) by a CHECK instruction to ensure complete and correct I/O termination. The actual ter-

mination of the I/O operation will cause an interruption, at which point the Supervisor will link to the SAM Posting and Error Retry (SPER) routine which runs with all interruptions disabled. Here the I/O completion code is posted into the DECB. This routine also performs various post-I/O functions, such as adjusting magnetic tape block counts. In addition, the error retry routines are incorporated into the SPER routine. The CHECK macro instruction, issued by the user at some point after his READ or WRITE, will test the indicators set by SPER.

If the DECB indicated unit exception, the Check routine invokes Mainline EOV, which in turn uses Tape Input EOV, Tape Output EOV, DA Input EOV, or DA Output EOV to complete EOV processing for tape or DA devices. When end of volume is in fact an end of data set condition, processing will involve setting an end of data set indication in the DCB. At this point Check is used to set up linkage to the user's end of data set routines if specified.

For DA output devices at EOV but not end of data set, the Extend routine is called to try to get more space on the current volume. If and when another DA output volume is necessary, Bump is called to mount/dismount the next volume and, in turn, Extend is used again to obtain space on that volume.

The Concatenation routine is used to link concatenated data sets.

EOV processing is transparent to the user; when end of volume but not end of data set occurs on input, outstanding reads will be automatically reissued by the Check routine upon return from EOV.

The end of data set indicator is recognized in the Check routine, and linkage is made to the user's end of data set routine, if specified.

In addition to Check, several other control routines are available and used in BSAM: Note, Point, Backspace, Control, and Force End of Volume.

To close his files, the user employs the CLOSE macro instruction which will link to Close Common. This routine resets fields in the DCB filled in by the access methods. For nonshared data sets, it does the necessary recataloging for volume extents.

Close Common will link to SAM Close. EOV is called to complete closing output tape or DA data sets. The Tape Output EOV routine will in turn utilize the label handling routines after it waits for outstanding I/O to quiesce. The SAM Close routine will release unused storage unless the user directs otherwise.

Another basic BSAM module is the Message Writer, which is called by the Open, Close and EOV routines to handle most messages and console communication, and do most ABEND processing.

See Section 6 of this PLM for routines specifically designed for BSAM.

SAM Communication Block: The SAM communication block is a table area used heavily by SAM Open, Close and EOV routines for passing parameters. See System Control Blocks PLM for a detailed description of this table.

MSAM OVERVIEW

Data Sets, Buffers, and Blocking Factors

The Multiple Sequential Access Method (MSAM) provides a fast and efficient mechanism for simultaneously driving several card readers, card punches, and printers under the control of a single user's task. Several data sets may be grouped together on any one device, allowing the user to process all of them under the same Data Control Block without opening and closing the DCB each time a data set with different characteristics is to be processed. Each of the separate data sets is referred to as a data group. Input data groups may be separated by control cards which consist of invalid EBCDIC characters in the first four columns and as many valid EBCDIC characters as required for control purposes. MSAM will recognize these control cards and notify the user that a control card has been read, allowing him to take whatever action is necessary. Output data groups on the card punch may be separated with special cards from the reader by specifying the COMBIN option in the DCB macro instruction, or they may be removed from the reader by the operator, who may be instructed to do so when a FINISH macro instruction is issued.

MSAM differs from other sequential access methods in that each MSAM I/O request of the system processes a buffer group of logical records, while each request issued by the other sequential access methods processes only a single physical record. Physical records are buffered by pages of virtual storage. MSAM processes a number of buffer pages based on an installation-provided parameter which is set in the symbolic device allocation table (SDAT), and which may vary for each device. Its value may be adjusted to provide optimum device utilization when the number of records which can be contained on N pages

will drive the device full speed for the maximum length of time between the two consecutive time slices.

The first 32 bytes of each buffer page are reserved for control information used by MSAM. The remaining portion of the page is packed with format-F or format-V logical records. Format-F logical records are packed in the buffer starting with the 33rd byte in the buffer. Format-V logical records are packed in the buffer starting with the 37th byte in the buffer, since four bytes must be reserved as control bytes (LLBB), as is the case with blocked, variable-length records.

The number of records per buffer page is restricted to a maximum of 100 on input and 200 on output. Depending on the size of the records, there may be fewer.

The size of an input buffer will be computed by adding to the 32 control bytes the smaller product of (a) 100 times the logical record length, or (b) multiplying by the logical record length the integral part of the result of dividing 4064 by the logical record length.

The size of an output buffer will be regulated by the following rules.

1. For fixed-length records, the number of bytes used for data will be the lower of (a) 200 times the logical record length, or (b) the product of multiplying by the logical record length the integral part of the result of dividing 4064 by the logical record length.

2. For variable-length records, the last record will have been placed in the buffer when (a) the record count reaches 200, or (b) the sum of the user-provided control bytes (LL) of each record in the buffer plus the next expected logical record length plus four is greater than 4064.

3. The buffer will be ended when form type-F is mounted on a printer, and a FORTRAN[1] control character is found indicating a skip to channel 1.

Schematic Description

The user's problem program initializes for an MSAM I/O operation by defining a data control block with the DCB macro instruction, which generates a common portion and an MSAM portion of the DCB. The user then issues an OPEN macro instruction which links to the Open Common routine.

Open Common completes the common portion of the DCB from the TDT JFCB, and then invokes MSAM Open to build a DEB and provide N buffer pages (where N is a constant in the SDAT set at system generation time). MSAM Open also provides N half-pages for IORCBs and a DBP page, which is used as a work area by MSAM Read/Write and DOMSAM. A DEB work page is also obtained to use as a save area for DOMSAM and to hold the N+1 DECBs. MSAM Open formats N IORCBs and N DECBs, and it checks the SDAT and the DCB for agreement and for valid options.

To set up online output devices, the user may issue a SETUR macro instruction. In the case of a print file, the SETUR routine may read the two system VIP data sets, SYSURS and SYSUCS, to obtain the parameters necessary for setting up the printer. SETUR will issue WTO macro instructions and possibly IOCAL SVCs to the I/O supervisor to achieve the desired setup of the device.

The user issues a GET macro instruction to obtain each card read from the card reader. Each GET macro instruction invokes the DOMSAM routine via type-1 linkage. If there are any records already in the buffer, DOMSAM passes the next sequential record to the user. If the buffer is empty, or if all the records in the buffer have already been processed, DOMSAM invokes MSAM Read/Write.

The user issues a PUT macro instruction to print each line on the on-line printer or punch each card on the on-line punch. Each PUT will cause a record to be placed in a succeeding location of a buffer page. DOMSAM keeps account of these records to determine when the last record has been placed in the buffer. At that time, it invokes MSAM Read/Write.

The MSAM Read/Write routine builds an IORCB and invokes the I/O Supervisor (IOS) via an IOCAL SVC. Each IORCB contains a list of CCWs for each record to be read or written. Each record read has a read CCW and a distinct feed, stacker select CCW associated with it. Each record written has essentially one CCW associated with it, for example, a punch, feed, select stacker CCW, or a print and space CCW. Additional control CCWs may be generated by MSAM at the beginning of the CCW list, such as skip to channel 1 on the printer. The IORCBs specify command and IORCB chaining and provide the address of the MSAM Posting routine.

---

[1]Control characters defined by American National Standard FORTRAN, ANSI X3.9 - 1966, hereinafter referred to as FORTRAN control characters (previously known as ASA or USASI control characters).

When the CCWs in an IORCB complete their execution, the currently running task program is interrupted, and MSAM Posting is given control from task monitor so that the necessary information may be stored in the DEB and in the DECB to inform DOMSAM of the I/O progress. If an I/O error has occurred, Posting will attempt the error retry procedures.

If intervention is required by the operator, Posting will record, in the DEB page, information about the IORCB returned by IOS in the ISA, and specify to the task monitor an asynchronous interruption routine to be given control when the device is transferred from the not-ready state to the ready state. A WTO macro instruction is issued indicating the required action, and control is returned to the task monitor, which returns control to the routine which was interrupted for the posting operation. The asynchronous routine is part of the MSAM Posting routine, but it has a separate entry point. When given control, it will reissue, from the point of failure, the CCWs in the IORCB which was posted by MSAM Posting.

By testing the return code from his GET or PUT macro instruction, the user can determine whether or not his operation has been completed. Before reissuing his incomplete GET or PUT, he is free to do other processing. Prior to reissuing a GET which returned an incomplete, the user should test the DECB pointed to by DCBCDE for completion; prior to reissuing a PUT which provides an incomplete return the user should test the DECB pointed to by DCBTDE for completion. If these DECBs are not complete and no further processing can be performed, the user may execute the AWAIT SVC in the DECB pointed to by DCBCDE or DCBTDE.

When the processing for the current data group is completed, the user may issue the FINISH macro instruction, which will invoke the MSAM Finish routine. On output, this routine will initiate type-1 linkage to DOMSAM, which may in turn invoke MSAM Read/Write for writing the last buffer on an output data group, and it will test the results of the write. It will wait for completion of all outstanding I/O requests for an input data group. Then it will notify the operator to remove the input or output data group from the device by a WTO, unless the user has indicated that such messages are to be suppressed.

If the operator has been requested to respond to the message by readying the device, the Finish routine notifies the task monitor to recognize an interruption when the affected device is changed from the not-ready to ready state. This interruption will give control to the Finish routine at its second entry point. The next time the Finish macro instruction is issued after this interruption is received, a return code other than incomplete is returned by the Finish routine to the user.

When no more data groups are to be processed by the task on the device at the present time, the CLOSE macro instruction is issued. The Close Common routine is invoked and clears all fields of the DCB completed by Open Common . It then invokes MSAM Close, which issues a FINISH macro instruction, waits for completion, and releases the storage areas obtained by MSAM Open.

MSAM Glossary

Tables 4, 5, 6, and 7 contain fields and/or flags used by MSAM which are frequently mentioned in the sections of this publication devoted to MSAM.

RTAM OVERVIEW

Activation of a terminal will cause an asynchronous interruption to be generated. The interruption will be fielded by the I/O Interruption Stacker routine in the resident supervisor and placed on the channel interruption processor queue. The Channel Interruption Processor determines that this is a terminal I/O interruption and passes it to the Terminal Communications Subprocessor, which is also resident.

This terminal control is all taking place in the resident supervisor, as the RTAM abbreviation suggests. For details or an overview of the RTAM access method, see the Resident Supervisor PLM, GY28-2012 or the System Logic Summary PLM, GC28-2009.

The Terminal Task Control routine, which describes internals for some of the MTT user commands and macros, is described in this PLM.

TAM OVERVIEW

Privileged programs are the only ones that may issue READ/WRITE macro instructions to directly call TAM Read/Write. Nonprivileged programs may only use the GATE I/O macro instructions that call an intermediate system's GATE routine that in turn calls TAM routines. The command system also invokes those GATE routines that link to TAM. All programs that use TAM routines either by a direct call or by the intermediate system's GATE routine are restricted in that they may only be used with specific communication terminals.

Table 4. DCB Table Fields and Flags (MSAM Section)

Field	Flag	Meaning
DCBINHMS	DCBINH	Inhibit message to operator to remove data group
DCBCOMB1	DCBCMB	Combine a reader on same 2540 as punch
DCBICE		Address of ICB named in SIR (0 = none)
DCBLRMAX		Maximum allowable logical record length
DCBLRC		Address of current logical record in buffer for input records, or next available buffer location for output records
DCBEAP		Address of end of current buffer
DCBPPT		Address of current buffer page
DCBRCX		Internal return code
DCBCNT		Logical record count
DCBCDE		Address of current DECB
DCBFDE		Address of first DECB in list
DCBLDE		Address of last DECB in list
DCBTDE		Address of DECB to be tested for completion on a PUT
DCBUDE		Address of user's copy of erring DECB
DCBFRMTP		SYSURS form type for printing
DCBSTRIK		UCS strike out code
DCBMSF1	DCBEOP	End-of-buffer processing needed
(MSAM flags)	DCBIOC	Read/Write already invoked
	DCBENT	Buffer priming to be performed
	DCBOVF	Format F new print page
	DCBELP	Last PUT issued was in locate mode
	DCBNLP	Previous locate mode PUT being processed
DCBMSF2	DCBPUR	Purge all I/O at CLOSE
(MSAM flags)	DCBSUR	SETUR in process
	DCBFIN	FINISH just issued
	DCBFIP	FINISH in progress
	DCBFT	First GET or PUT on a data group

Table 5. Some DBP Table Fields

Field	Meaning
DBPPRTRY	Printer retry counter
DBPPRDC	Printer data check counter
DBPFRMTP	SYSURS form type code
DEPFOLD	SYSURS UCS folding code
DBPSTRK2	SYSURS UCS strike out code

Table 6. Some DECB Table Fields (CHADEC)

Field	Flag	Meaning
DECECB (completion code)	DECEC0	Read/Write request code
	DECEC1	Normal Completion
	DECEC2	Complete with error
	DECEC3	Intercepted
	DECEC4	Wait
DECLEN		Data area length
DECCSW		Channel status word

Table 7. Some DEB Table Fields

Fields	Flag	Meaning
DEBIOC		Number of outstanding IORCBs
DEBNF	DEBNF1	Unrecoverable I/O error
	DEBNF2	Permanent I/O error
DEBCLS		Storage protection class of DCB

During execution the OPEN macro instruction provides linkage to the Open Common routine. Open Common locates the corresponding JFCB for the data set. Open Common links to TAM Open to continue special open functions, and then TAM Open returns to Open Common which sets bits in the DCB and the JFCB to say that the DCB is open. A counter in the JFCB is updated to indicate the number of DCBs that are open.

During TAM Open, one page of storage is allocated and pointers are set up between this page and other control blocks. Part of this page is reserved for the DEB, which is partially completed during TAM Open with terminal information that was stored in the the symbolic device allocation table (SDAT). The remainder of this page is reserved for the terminal operational status table (TOS) which includes the IORCB. The TOS is used as a work area during TAM Read/Write in order to complete the IORCB. Since the SDAT contains current information about the terminals, TAM Open increments the SDAT DCB open count for this terminal by one.

To accomplish a read/write function, the corresponding TAM GATRD/GATWR macro instruction is required in the nonprivileged program while READ/WRITE macro instructions may be used in a privileged program. During assembly this generates a DECB that will be used to store the I/O status of this operation. During execution from the terminal information (terminal type and model code stored in the DEB during TAM Open), and from the type option (stored in the DECB), TAM Read/Write begins a table search. This search is through three internal tables (unit type, terminal library, and terminal control program) to locate a prestored channel program generator (CPG) for the terminal. This CPG is made up of channel command word generators (CCWGs) that use the work area in the TOS to build channel command words (CCWs) and then move them into the IORCB. The I/O buffer area is also completed in the IORCB. The CCW list is executed by issuing an IOCAL SVC. This passes the IORCB to the I/O supervisor to execute the CCWs. At the completion of the I/O operation an interruption occurs. The IORCB and the complete I/O status information are stored in the interruption storage area (ISA located at a fixed location of segment 0, page 0).

TAM Posting processes this I/O interruption by decoding the interruption data. The CCW list that was executed during TAM Read/Write is traced through again to locate read CCWs. The Data In processor assures that the user read area is available and translates and moves the data to this area. TAM Posting does not issue an ABEND upon noting exception or error conditions, but only posts this exception information in the DECB. It is the user's responsibility to verify correct operations with a CHECK macro instruction so the user's program may continue. However, if error conditions occur, the user's SYNAD routine may be entered where the address of this routine is pointed to by the DCB.

When the user's I/O operations with the terminal are completed and a close is issued, the CLOSE macro instruction links to the Close Common routine. Close Common closes the DCB by restoring it to the original status. Close Common then links to TAM Close to continue the special close functions and then TAM Close returns to Close Common to reset bits in the DCB and the JFCB to indicate a closed DCB.

TAM Close frees the storage page allocated during TAM Open and resets the required pointers.

The SDAT DCB open count for this terminal is decremented by 1. For LOGOFF at all terminals, the disable/enable logoff function imbedded in CLOSE is also required. A recursive call flag prevents a recursive loop between TAM Close and ABEND.

IOREQ OVERVIEW

Privileged programs are the only ones that may use the IOREQ routines for unit record equipment or using SDAT. The IOREQ programs access data from any private I/O device. The data is accessed at the channel command word level. A data set organization of RX (IOREQ facility being used) must be specified in the DDEF command.

When OPEN is issued, the Open Common routine opens the DCB and locates the corresponding JFCB for the data set. Open Common links to IOR Open to continue special open functions and then IOR Open returns to Open Common which sets bits in the DCB and the JFCB to indicate the DCB is open.

During this IOR OPEN, tests are made to verify that: (1) the user's privilege class is E for unit record equipment, (2) IOREQ is specified in the DDEF command, (3) IOREQ is allowed on the device, and (4) this device is a private volume. Storage is allocated for the DEB and the IORCB control blocks with data type code information moved from the JFCB to the DEB. A final check assures that the user is privileged, if access to the volume is privileged.

To accomplish a read/write, the IOREQ macro instruction is required in the user's program. This generates, during assembly, a DECB table that will be used to store the I/O status of this operation. The I/O operation is specified by VCCW macro instructions specified in the user's program. These VCCW macro instructions are used by IOREQ to generate a list of CCWs to control the I/O activity. IOREQ places this list of CCWs in the IORCB. If buffering is requested by the user, space is allocated in the IORCB for the buffer area. If buffering is not requested by the user, space is allocated in the IORCB for page list entries to connect the CCWs to the I/O areas. The CCWs are executed by issuing an IOCAL SVC. This passes the IORCB to the I/O supervisor (IOS) to execute the CCWs. At the completion of the IOREQ I/O operation an interruption occurs, and the IORCB and the complete I/O status information are stored in the interruption storage area (ISA, located at a fixed location of segment 0, page 0).

IOREQ Posting processes this I/O interruption by analyzing the interruption data with all other task interruptions masked off. It then posts the normal or abnormal completion code in the DECB allowing the Check routine to later take action based on

these codes. IOREQ Posting does not contain any error recovery routines. The CHECK macro instruction must be used to ensure the completion of the I/O operation and to detect errors or exception conditions. If the I/O operation is successful, the program resumes execution at the instruction after the CHECK macro instruction. If the I/O operation results in an unusual condition, the check of the DECB associated with this IOREQ causes control to be given to the user's SYNAD routine specified in his DCB. If multiple IOREQs are issued before a check of the first IOREQ is made, and one of these IOREQs generates an error, the subsequent IOREQs will be intercepted by IOS. It is the user's responsibility to reissue any IOREQ following the error-causing IOREQ.

The CHECK macro instructions must also be issued in the same order in which the associated IOREQ macro instructions were issued.

When the user's I/O operations with the device are completed, the CLOSE macro instruction links to the Close Common routine. Close Common then links to IOR Close to continue the special close function, and then IOR Close returns to Close Common to reset bits in the DCB and the JFCB to indicate a closed DCB.

This IOR Close waits until all outstanding DECBs have been completed and then frees the storage allocated during IOR Open.

COMMON PROCESSING

The following routine is common to all access methods OPEN processing.

Open Common Routine (CZCLA)

The Open Common routine, called by the OPEN macro instruction, performs those open functions common to all access methods:

- It checks for DCB error conditions, and ABENDs if any exist.
- It uses the GETMAIN macro instruction to acquire a one-page work area, and passes the address of the work area as part of a parameter list when it links to the access-dependent Open routines.
- It places the open options in the DCB for reference by the access-dependent open routines.
- If necessary, it issues a call to FINDJFCB to find the JFCB associated with the data definition name (ddname) of the DCB.
- It fills in certain defaulted DCB fields with information from the corresponding fields in the JFCB. It keeps track of the DCB fields so modified, so that at CLOSE time the DCB can be restored to its pre-OPEN status.

In addition to these operations common to all access methods, Open Common automatically catalogs all VAM data sets.

Open Common links to the appropriate routine for access-dependent open processing (Chart AA).

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged routine, public.

Entry Point: CZCLAO -- Entered by type-1 or type-2 linkage.

Input: Register 1 contains the address of the CHAGSM table. CHAGSM (the general services macro table), built by the expansion of the OPEN macro instruction, consists of one doubleword entry for each DCB to be opened.

Data References: CHADCB, CHATDT, CHAGSM, CHAISA.

Modules Called:
FINDJFCB (CZAEB) -- Find JFCB.

SAM Open (CZCWO) -- SAM Open.

TAM Open (CZCYA) -- TAM Open.

MSAM Open (CZCMC) -- MSAM Open.

Open VAM (CZCOA) -- VAM Open.

IOR Open (CZCSC) -- IOR Open.

VMA (CZCGA) -- Get virtual storage.

Addcat (CZCFA) -- Catalogs all VAM data sets.

Search SDST (CZCQE) -- Search shared data set table.

Read/Write (CZCPE) -- Index sequential read/write.

Exits:
Normal -- Return to calling program.

Error -- ABEND macro instruction.

Operation: Open Common provides DCB addressability and checks for valid DCB identifier and nonzero ddname. If an error condition exists, the task will ABEND.

One of the primary functions of Open is to find the JFCB for the data set being opened. If the data set is concatenated, the address of the JFCB is picked up from DCBCON; otherwise, the routine examines the TDT for the JFCB address. If still not found, the address of the JFCB is obtained through a call to the FINDJFCB routine.

Following the call to FINDJFCB, VAM data sets are automatically cataloged by calling Addcat.

A user with read-only access is allowed to open a VAM data set even though he has specified it as modifiable for his purposes (OUTPUT, EDIT, or other options). This allows him to use the data set locking feature; he will be prevented from modifying the data set by the VAM output routines. An existing read-only data set not of VAM organization and specified by the user with other than the INPUT option will result in an ABEND.

Open Common will turn on the concatenated system flag in the DCB if the JFCB describes a concatenated data set.

The zero DCB fields are filled in with corresponding entries from the JFCB, enabling the user to specify many data set characteristics and handling options for

this run that were not specified during assembly.

Open Common gets a page of storage which is used by the BSAM for IORCBs, and by fence straddlers and VAM for save areas, then links to the appropriate access-dependent open routines.

Upon return from the access dependent routine, Open Common tests for other DCBs to be opened. The entire procedure is repeated for each DCB and when all DCBs have been opened, control is returned to the calling routine.

SAM OPEN PROCESSING

The following routines are common to SAM processing.

SAM Open Mainline Routine (CZCWO)

This routine performs opening functions common to sequential access methods. It branches to the Open Tape or Open DA routines to have the open processing completed for magnetic tape or direct access devices respectively (Chart AB).

Attributes: Reentrant, resident in virtual storage, closed, privileged.

Entry Point: CZCWO1 -- Entered only by type-1 linkage.

Input: When this routine is entered, register 1 contains the address of the following three word parameter list:

Word 1 -- Address of DCB being opened.

Word 2 -- Address of associated JFCB.

Word 3 -- Address of work area for building IORCBs.

The PSECT of CZCWO contains the SAM communication block (CHASCB), three temporary control blocks - a DCB, a DEB and a DECB which are used by the label processors for reading or writing tape labels, and a parameter area for reading and writing format-1 DSCBs.

Data References: CHADCB, CHADEB, CHASCB, CHATDT, CHADEC.

Modules Called:
DA Open (CZCWD1) -- Open direct access.

Tape Open (CZCWT1) -- Open tape.

Mainline EOV (CZCXE1) -- Write EOV trailer and header labels when EOV encountered during header label processing.

User Prompter (CZATJ1) -- Write a warning message.

VMA (CZCGA2, CZCHA2) -- Get virtual storage.

VMA (CZCHA3) -- Free virtual storage.

Volume Sequence Convert (CZCWV1) -- Volume address conversion.

Exits:
Normal -- Return to calling program.

Error -- Issue ABEND.

Operation: The SAM Open Mainline routine initializes the SAM communication block (CHASCB). If a DCB is currently opened on the JFCB, the task is abnormally terminated. Open options are checked against the JFCB disposition parameter to see if a data set with a disposition of NEW is opened for input. If it is, and the task is nonconversational, the task is abnormally terminated; for conversational mode tasks, the routine gives the user a warning and the option to continue.

SAM Open checks to make sure that the data set has a mounted volume, and that the proper volume is mounted. The routine also checks to make sure no reading will be performed on output data sets, or writing on input data sets.

The main function performed by SAM Open is the building of the data extent block in privileged storage so that it cannot be destroyed or changed by the user program.

If the device assigned to the data set is a magnetic tape or direct access device, control is given to Tape Open or DA Open respectively. These routines, in turn, call the proper DEB building routine.

If Tape Open or DA Open had been given control, any storage dynamically obtained by either routine is released by calling FREEMAIN, and normal return is made to the user.

Whenever Tape Open or DA Open encounters errors, it posts an abnormal condition code in the SCB and terminates via ABEND.

In case an end-of-volume condition occurs while Tape Open is writing the header, SAM Mainline will call Mainline EOV to end the present tape with an EOV trailer label and write the header on a new tape.

The block size of ASCII format tapes is checked for minimum (18 bytes) and maximum (2048 bytes) length.

QOPEN is bypassed if neither GET nor PUT is indicated in the macro field of the DCB. If the data set being opened is a QSAM data set, then QOPEN, a section of SAM Open Mainline, is entered to perform those functions unique to a QSAM data set. If blocked records were indicated in the DCB and the blocksize is zero, an ABEND exit is taken. Otherwise, the block size is set equal to the maximum logical record length. Then, the number of buffers to be obtained for the data set must be determined. If the data set is opened for UPDATE, and CNTRL is specified in the DCB, only one buffer is needed. If the data set is opened for RDBACK, and if the record format is variable, three buffers must be obtained. Otherwise, two buffers must be obtained. The storage for buffers is obtained by issuing a GETMAIN macro instruction which returns the address of the area obtained in register 1. This address is saved in the DCB. If two or three buffers are needed, their addresses are calculated and stored in the DCB. The protection class of the area will be the same as that of the DCB. For data sets using only one buffer, the DCB field indicating the maximum number of reads or writes which may be done before a check is set to one, and for all others it is set to two. Then the same procedure described above is followed to obtain storage for QSAM's QWK work area. (See the QSAM section of this publication for a description of the QWK work area.)

The V-cons for the entry points CZCSAA, CZCSAB, CZCSAX, and CZCSAS are set into the DCB. If the data set is opened for output, the GET V-con field in the DCB (DCBGTV) is set to one. Otherwise, the PUT V-con field (DCBPTV) is set to one. This is to insure that no GETs are issued on an output data set, and no PUTs issued on an input data set.

Tape Open Routine (CZCWT)

The Tape Open routine completes the open processing for a sequentially organized data set on magnetic tape. It builds a data extent block (DEB), uses the appropriate label processor to process tape labels, and completes the tape recording information fields in the DCB. (See Chart AC.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged.

Entry Points: CZCWT1 -- Entered with type-1 linkage.

Input: Register 1 contains the address of the SAM communication block (CHASCB). Note that the SCB contains pointers to a DCB, DEB, and a DECB in the SAM Open PSECT which are used for reading or writing tape labels, as well as pointers to the actual DCB being opened and its associated JFCB.

Data References: CHADCB, CHATDT, CHASCB, CHADEB, CHAISA.

Modules Called:
Control (CZCRB) -- Magnetic tape positioning.

Bump (CZCAB) -- Request and verify mount of new volume.

VMA (CZCGA) -- Get virtual storage.

LVPRV (CZCJL) -- Leave privileged state.

Build Common DEB (CZCWB) -- Build the common portion of a DEB.

Tape Data Set Label (CZCWY) -- Tape label processor.

Tape Positioning (CZCWP) -- Position tape.

User Prompter (CZCTJ) -- Inform user of error.

Exits:
Normal -- Return to calling program.

Error -- Via ABEND macro instruction.

Operation: If the tape is labeled, GETMAIN is called to get an area of virtual storage for label buffers. If the routine determines from the SCB that the correct volume for the data set is not mounted, the Bump routine is called to mount the proper volume.

The Build Common DEB routine is called to build the common portion of the DEB for the data set being opened. This portion of the DEB is copied into the "temporary" DEB pointed to by the SAM communication block for use in processing labels.

The tape volume is positioned by calling the Tape Positioning routine; the volume labels are then written or read via Tape Data Set Label. If the OPEN option is INPUT, INOUT, or RDBACK, or if the JFCB indicates MOD, the data set labels are processed as input; otherwise the label is processed as output.

The labels are processed unless the JFCB indicates no labels. If the user requests it, he is given control at this time, through the DCB exit, to modify the DCB.

The tape recording fields in the user's DCB are completed. The User Prompter routine is called to request directions from the user if there are incompatibilities in the user's specifications for tape recording in his DCB; for example, recording

density of 200 bits per inch is incompatible with 9-track magnetic tape. Should the user not supply a satisfactory solution to the problem, Tape Open will effect an abnormal end with an ABEND.

DA Open Routine (CZCWD)

The DA Open (Direct Access Open) routine completes the open processing for a SAM data set on a direct access device. It builds a data extent block (DEB), processes DSCBs and user labels, sets necessary fields in the DEB, and makes sure the proper data set volume is mounted. (See Chart AD.)

Attributes: Reentrant, resides in virtual storage, closed, read-only, privileged.

Entry Point: CZCWD1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block (CHASCB).

Data References: CHATDT, CHADSC, CHADCB, CHADEB, CHASDA, CHASCB, CHAISA.

Modules Called:
Bump (CZCAB) -- Request and verify mounting of new volume.

Obtain/retain (CZCFO) -- Obtain DA user label and retain DA user label.

VMA (CZCGA) - Get virtual storage.

LVPRV (CZCJL) -- Leave privileged state.

Build DA DEB (CZCWL) -- Build direct access DEB.

User Prompter (CZCTJ) -- Inform user of error.

Read Format-3 DSCBs (CZCWR) -- Read and chain format-3 DSCBs.

DA Input Label (CZCXN) -- Direct access input label processor.

DA Output Label (CZCXU) -- Direct access output label processor.

Exits:
Normal -- Return to the calling routine.

Error -- Via ABEND.

Operation: If the volume to be processed is not the one which is mounted, DA Open calls the Bump routine to mount the proper volume.

The Obtain routine is called to get the format-1 DSCB for the data set. If the integrity bit is on in the DSCB, a PRMPT macro is issued to ask the user if he wants to continue. The integrity bit is set on when the data set is opened, and set off at EOV or CLOSE by Set DSCB. Therefore, if the integrity bit is already on during the open process, the data set was previously opened but never closed.

If the data is being opened for output and has a disposition of OLD, and the expiration date in the DSCB has not been reached, a PRMPT macro is issued to ask the user if he wants to write on the unexpired data set. The Retain routine is used to write the format-1 DSCB on the volume. For old data sets, zero DCB fields are completed with fields from the format-1 DSCB, and the expiration date is stored in the JFCB from the DSCB field.

Should the data set have format-3 DSCBs, they are read by calling the Read Format-3 DSCB routine. Then since all extents are known, the Build DA DEB routine is called to build a direct access DEB.

If the user labels are specified, GETMAIN is used to get buffer space for label processing.

DA Open sets on the integrity bit in the format-1 DSCB when processing a volume with a data set opened for OUTPUT, OUTIN, or INOUT. As processing of each volume is completed, the integrity bit is reset by other BSAM routines.

The DEB is set to point to the first data record except for a data set with disposition MOD. In the latter case, the DEB is set to point to the end of the last data record as indicated in the format-1 DSCB.

The user labels are then read or written using either the DA Input User Label processor or the DA Output User Label Processor.

DEB PROCESSING

The following routines are used to build or modify the data extent blocks.

Build Common DEB Routine (CZCWB)

The Build Common DEB routine may be entered to perform either of two functions. It can obtain virtual storage to create a DEB and initialize the DEB's common portion. Alternatively, Build Common DEB may be called to modify the common portion of an existing DEB. (See Chart AE.)

Attributes: Reentrant, resident in virtual storage, closed, privileged.

Entry Point: CZCWB1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block (CHASCB).

Data References: CHASCB, CHADCB, CHADEB, CHATDT, CHASDA.

Modules Called:
VMA (CZCGA) -- Get virtual storage.

User Prompter (CZCTJ) -- Issue message to user.

Exits:
Normal -- Return to calling routine.

Error -- Via ABEND.

Operation: When the routine is called to build the common portions of the DEB, storage is obtained via GETMAIN and all the fields in the common portions of the CHADEB which can be filled from the CHADCB, CHASDA, and CHASCB are initialized. The remaining fields of the CHADEB are zeros.

If the routine is called to reinitialize fields for the appropriate CHASDA, Build Common DEB reinitializes, in both the CHADEB and the temporary access method CHADEB, those fields originally obtained from the CHASDA and DEVOL.

This routine abnormally terminates if the size of the CHADEB equals zero.

Build DA DEB Routine (CZCWL)

The Build DA DEB routine creates a data extent block (DEB) for the first volume of a data set on a direct access device. Additionally, it can add extents to an existing DEB, or build a new DEB for a multivolume data set when the extents indicated in the old DEB are obsolete. (See Chart AF.)

Attributes: Reentrant, resident in virtual storage, closed, privileged.

Entry Point: CZCWL1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block (CHASCB). The CHASCB will contain a pointer to the current DEB, a pointer to a chain or format-1 and/or format-3 DSCBs, and an indication of whether the routine is to construct or extend a DA DEB. The routine assumes that the chained DSCBs are in virtual storage.

Data References: CHASCB, CHADEB, CHADSC, CHADCB, CHATDT, CHASDA.

Modules Called:
VMA (CZCGA) -- Get virtual storage, free virtual storage.

Point (CZCRM) -- Logically reposition data set.

Build Common DEB (CZCWB) -- Build and modify the common portion of the DEB.

User Prompter (CZCTJ) -- Issue message to user.

Exits:
Normal -- Return to caller.

Error -- Via ABEND.

Operation: This routine first calculates the actual size (in bytes) of the DEB. The size is the sum of: the number of bytes in the common portion of the DEB, four times the number of channel programs (DCBNCP) less one, the number of bytes in the fixed length direct access portion of the DEB, and sixteen times the number of extents to be contained in the DEB. The number of extents is determined by searching the DSCBs until either a null extent type code or a null CCHHR chain address is found.

If the data set has not been opened, the Build Common DEB routine is called to construct and initialize the common portion of a DEB. The extents (addresses) are then stored in the DEB from the indication of extents in the DSCB chain. Control is then returned to the calling routine.

If the data set is open, this routine has been entered with a DEB already in existence. Therefore, a new DEB must be generated and the present extents entered in it.

GETMAIN is called to obtain virtual storage for the new DEB. If the next extent to be processed is zero, the fixed portion of the old DEB is moved into the new DEB, and the Build Common DEB routine is called to modify the volume and device fields in the DEB. Should the next extent to be processed be non-zero, the common and fixed direct access portions of the old DEB are copied into the new DEB, and Build Common DEB is called for the volume and device field modifications of the new DEB. The DEB fields for next I/O, last I/O, and last write addresses are initialized. If the JFCB indicates DISP=MOD, Point is called to position logically to the end of the data set. In either case FREEMAIN is called to release the storage of the old DEB, and extents are added to the new DEB on the basis of those extents currently indicated in the DSCB chain. Control is returned to the calling routine.

This routine abnormally terminates if:

1. The DSCB extents are not equal to the number of calculated extents.

2. The DSCB extents are not numbered in consecutive order.

DSCB PROCESSING

The following routines concern SAM DSCB processing.

Read Format-3 DSCBs Routine (CZCWR)

The Read Format-3 DSCBs routine causes all format-3 DSCBs associated with one volume of a data set to be read into virtual storage and chained together. (See Chart AG.)

Attributes: Reentrant, resident in virtual storage, privileged.

Entry Point: CZCWR1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block (CHASCB).

Data References: CHASCB, CHADSC, CHADEB.

Modules Called:
Obtain/Retain (CZCFO) -- Obtain DA user label.

VMA (CZCGA) -- Get virtual storage.

User Prompter (CZCTJ) -- Issue message to user.

Exits:
Normal -- Return to the calling routine

Error -- Via ABEND.

Operation: It is necessary to compute the amount of virtual storage needed for reading the format-3 DSCBs. The correct number of bytes is calculated and stored in SCBF3Z. GETMAIN is called to get that calculated number of bytes of virtual storage.

The DSCBs are then read into the storage which GETMAIN supplied. The Obtain routine is used to read each DSCB. The DSCBs are chained together. Where there are no more DSCBs to be read, control is returned to the calling routine.

Set DSCB Routine (CZCXS)

The Set DSCB routine updates the information in format-1 DSCBs, turns off the integrity bit in the format-1 DSCB, and writes a file mark on the DA output volume. (See Chart AH.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged.

Entry Point: CZCXS1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block (CHASCB).

Data References: CHADEB, CHADCB, CHADSC, CHASCB, CHASDA, CHATDT, CHADEC.

Modules Called:
Read/Write (CZCRA) -- BSAM read/write.

Obtain/Retain (CZCFO) -- Obtain DA user label and retain DA user label.

FULREL (CZCRS) -- Convert full DA address to relative address.

User Prompter (CZCTJ) -- Communicate with user.

Exits:
Normal -- Return to calling program.

Error -- Via ABEND.

Operation: If the SCBRF1 flag is on, the format-1 DSCB is read, the integrity bit is set off, and the DSCB is written.

Otherwise, the format-1 DSCB is read via OBTAIN, and the DCB type fields in the format-1 DSCB, volume sequence number, and the last volume bit are written. The Last Record pointer is set to point to the last record written, the bytes left on the track are stored in DSCLRD, the integrity bit is set off, and the format-1 DSCB is rewritten via RETAIN to reflect the above. A file mark is placed following the last record that was written on the volume and the SCBFLG, which contains the SCBRF1 flag, is set to zero.

MSAM PROCESSING

The following routines are used with MSAM processing.

MSAM Open Routine (CZCMC)

The MSAM Open routine edits the DCB and SDAT for valid options and combinations of options, and constructs control tables (DEB, IORCBs, and DECBs) and work areas (DEB page and buffer pages) in virtual storage for use by the multiple sequential access method. (See Chart AI.)

Attributes: Reentrant, read-only, public, privileged, system, nonrecursive.

Entry Point: CZCMC1 -- Entered by type-1 linkage from Open Common when the DSORG field of the DCB specifies MS.

Input: When this routine is entered, register 1 contains the address of the following parameter list:

Word 1 -- Address of the DCB.

Word 2 -- Address of the TDT JFCB.

Data References: CHADCB, CHASDA, CHATDT, CHADEB, CHADEC, CHAIOR, CHAICB, CHADBP.

Modules Called:
VMA (CZCGA) -- Get virtual storage.

CKCLS (CEAQ4) -- Check storage protection class.

Exits:
Normal -- Return to Open Common.

Error -- Via ABEND.

Operation: When MSAM Open is entered, register 1 contains the address of a parameter list which contains a full word pointing to the DCB and a full word pointing to the TDT JFCB. MSAM Open accesses these two addresses, and obtains the address of the SDAT from the TDT. If a DCB has been opened previously for this data set, the task is abnormally terminated unless it is a remote job entry (RJE) task. If it is RJE, the task is abnormally terminated if more than one previous DCB has been opened. (The JFCB indicates a previously opened DCB.)

There are two tables (TBDD1B and TBDD2B) in MSAM Open for each of the device dependent parameter fields in the DCB. These tables contain the allowable parameters and the default parameters for the particular device dependent fields. If the value of a device dependent field does not match any of the allowable parameters, the default parameter is stored in the field. Checks are made on fields of the DCB and SDAT for valid options and combinations of options. Any invalid condition causes abnormal termination of the task with the appropriate message displayed on SYSOUT via the ABEND macro instruction.

The value of N, the maximum number of allowable IORCBs, is obtained from the SDAT and a check is made to determine the storage protection class of the DCB. If the DCB is Class A (user read-write), a GETMAIN macro instruction is issued to obtain (N+3)/2 contiguous pages of Class B (user read-only) virtual storage. These pages will be used for the DEB page and the (N+1)/2 IORCB pages which cannot be of Class A storage. (See Figure 2 and Figure 3.) A pointer to the first page obtained

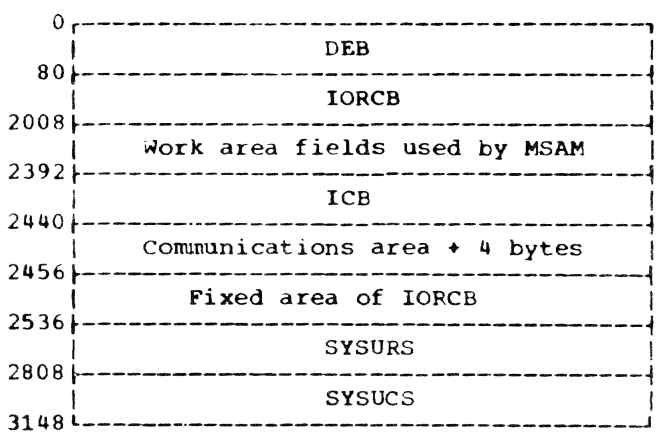


Figure 2. DEB Page Layout

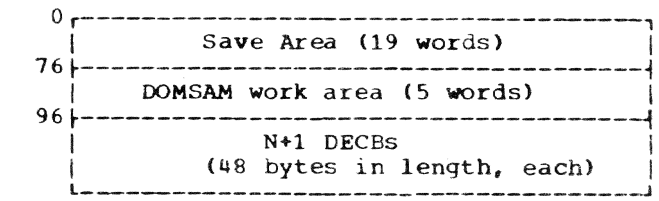


Figure 3. DEB Work Page Layout

is stored in the DCB. A second GETMAIN is then issued to obtain N+1 contiguous pages of Class A virtual storage for the DEB work page and the N buffer pages which must be of the same protection class as the DCB.

If the DCB is Class B (user read-only) or Class C (user inaccessible), a single GETMAIN is issued for (3N+5)/2 contiguous pages of virtual storage of the same protection class as the DCB. In both cases the same number of pages are obtained, although in the first case the two groups of pages will not necessarily be contiguous. Except for the case where the DEB and IORCB pages cannot be Class A, all pages obtained are of the same class as the DCB.

Fields in the DEB may now be initialized, and pointers set to the other control tables. For an RJE task, DEB pointers in the TDT JFCB are enchained if a previous DEB has been created during the task. A skeleton ICB is built to specify an attention-type interruption, with pointers to the DCB and associated communication area. The fixed area of an IORCB is built, and additional fields in the DEB page are initialized for use by the other MSAM modules. If RJE, flags are set in the IORCB and the DEB.

Next, N skeleton DECBs are built in the DEB work page following the 19 word save area of DOMSAM, and a pointer to the first DECB is set into the DCB. To initialize

for looping, the DCB pointer to the current DECB is initialized to the address of the first DECB.

For fixed record format, the number of logical records in the buffer is then computed as the buffer size divided by the logical record length, where the buffer size is equal to 4096 minus the number of control bytes (currently 32). If this count of logical records is greater than 100 on input, it is set to 100, and if it is greater than 200 on output, it is set to 200. For variable record format, the number of logical records in the buffer is initialized at zero. If RJE, logical record size is stored in the DEB work page area. The following fields are calculated and saved: the maximum record count for RJE PUT (4048/(LRECL+2+15))/2*2; the maximum record count for the RJE first PUT of the first IORCB with machine control characters (4040/(LRECL+2+15))/2*2; and the maximum size of the PUT logical record (LRECL+1+15)/8*8. After calculation of logical record size and count limit, a loop is performed to initialize N skeleton DECBs.

Now the N IORCBs are initialized in their half pages. The fixed area of the first IORCB (at the beginning of the first full page beyond the DEB page) is built. Since each of the other N-1 IORCBs is to be built in the same manner as the first, the fixed area of the first IORCB is moved to these other N-1 IORCBs, which are located at successive half-page boundaries following the first IORCB.

Next, the fields in the MSAM portion of the DCB are initialized. The DCB pointers to the last, the current, and the user's DECBs are set, and the fields indicating the number of logical records and the return code are both set to zero. A flag is set in the DCB to indicate that the next GET or PUT issued will be the first on this data group.

The DCB macro field is then tested. If it specifies PUT, the DOMSAM PUT VCON and RCON are set into their respective fields in the DCB, and unused VCONs in the DCB are set to full words of hexadecimal Fs. If the records are fixed length, the address of the current logical record is computed as the beginning-of-buffer address plus the number of control bytes. If the records are variable length, a test is made for an RJE task. If RJE, the address of the current logical record is set to the beginning-of-buffer address plus the number of control bytes plus 9. If not RJE, it is set to the beginning-of-buffer address plus the number of control bytes plus 4, and the block control bytes (LLBB) are initialized to X'0004'C'bb'.

If the DCB macro field specifies GET, the DOMSAM GET VCON and RCON are set into their respective fields in the DCB, and unused VCONs are set to full words of hexadecimal Fs.

Control is then returned to Open Common. There is no return code.

SETUR Routine (CZCMD)

The Set Unit Record (SETUR) routine specifies the unit record configuration for a local or remote printer or a card punch. It is called as the result of a user-initiated SETUR macro instruction, indicating how a device is to be set up for a job. If the device is not already correctly set up, the SETUR routine requests an operator to set the device as specified by the macro and sends the user return codes indicating the results. (See Chart AJ.)

Attributes: Privileged, reentrant, read-only, public, system, nonrecursive.

Entry Points:
CZCMD1 -- Primary entry point entered with type-1 or type-2 linkage.

CZCMD2 -- Asynchronous entry point entered with type-2 linkage.

CZCMD3 -- Synchronous entry point entered with type-1 linkage.

Input:
For entry at CZCMD1, register 0 contains the address of the unit record device setup parameter and register 1 contains the DCB address.

For entry at CZCMD2, register 1 contains the ICB address.

For entry at CZCMD3, the ISA contains the IORCB.

Data References: CHADCB, CHASDA, CHADEB, CHAIOR, CHAICB, CHAISA, CHADBP.

Modules Called:
WTO (CZABQ) -- Write message to operator on console typewriter.

SIR (CZCJS) -- Specify interruption routine.

DIR (CZCJD) -- Delete interruption routine.

Open (CZCLA) -- Open a data control block.

Find (CZCOJ) -- Find a member of a VPAM data set.

Read (CZCPE) -- Read a VISAM record.

Close (CZCLB) -- Close a data control block.

Reset (CEAAH) -- Reenable a device after I/O error.

Exits:
Normal -- For return from CZCMD1 register 15 contains one of the following codes:

'00'	Completed successfully.
'04'	Incomplete.
'08'	Unrecoverable I/O error.
'0C'	Bad parameter for SYSURS key.
'10'	Invalid SYSUCS key specified by SYSURS.
'14'	Intervention required on RJE device.

For return from CZCMD2 or CZCMD3 register 15 contains '00'.

Error -- Abnormal termination via the ABEND macro instruction.

Operation: The operation at the three entry points is:

MAIN ENTRY AT CZCMD1: SETUR abnormally terminates if the DEB or DCB is invalid, or if the DCB has not been opened.

Since the SETUR macro instruction is issued repetively by the user until he receives a return code other than 4 (incomplete), the SETUR routine must determine conditions existing each time it is invoked. The internal return code in DCBBCX, in combination with the SETUR-in-progress switch (DCBSUR), determines the line of processing to be followed upon entry. DCBSUR is set on when SETUR is first called for a device; it is set off when SETUR has completed its processing, or when the invoking routine wishes to notify SETUR to stop its processing when it is next given control. Processing for the various DCBSUR, DCBRCX combinations is as follows:

DCBSUR off, DCBRCX less than 100: If the device is other than a card punch or a printer (or, if an RJE task, other than a printer), SETUR exits to the caller with a return code of normal completion.

For a card punch: (DCBSUR off, DCBRCX less than 100). If the setup parameter card form number is the same as the SDAT form number, normal ending procedures are followed. The return code is set for normal completion, DCBSUR is set off, and control returns to the calling routine.

If the numbers are not equal, a message is sent to the operator via WTO requesting him to mount the desired form, CZCMD2 is specified as the asynchronous entry point and SETUR returns to the caller with a return code for incomplete while operator response is awaited.

For a printer (or, if RJE, remote printer): (DCBSUR off, DCBRCX less than 100). If the setup parameter key is the same as the SDAT URS key, the desired configuration is already set, so normal ending procedures are followed as with the card punch.

Otherwise, the DCB for the VPAM data set containing the member $SYSURS is opened, and the four-line SYSURS record is read according to the key given in the setup parameter. If an error occurs on a READ, SETUR saves the error information in the MSAM DCB, closes the SYSURS file, turns DCBSUR off, and exits to the caller with a return code indicating either "invalid SYSURS key" or "unrecoverable error". When the SYSURS record has been read successfully, the SYSURS DCB is closed and checks are made for appropriate printing specifications and valid SYSURS parameters. SETUR abnormally terminates if the conditions checked for are not met.

SETUR now checks that the required print form, carriage tape, print chain/train and density are now being used. If any of these need to be changed, an appropriate message is sent to the operator via the WTO macro instruction, or, if an RJE task, to the remote operator via an IOCAL macro instruction, after building a special IORCB. CZCMD2 is specified as the asynchronous entry point by a SIR macro instruction unless the task is RJE, in which case this entry point will have been specified as part of BULKIO initialization. SETUR returns an 'incomplete' indication while local or remote operator response is awaited.

If no such changes are necessary, the SYSURS form type value is saved in the DCB, and SETUR tests for use of the Universal Character Set (UCS) feature.

For UCS printing, SETUR abnormally terminates if the folding code is invalid. If the SDAT and SYSURS values for the UCS key do not match, the DCB for the VPAM data set containing the VIP member $SYSUCS is opened, and the 5-line SYSUCS record is READ according to the key given in SYSURS. If no error occurred on the read operations, the SYSUCS DCB is closed, an IORCB is built to load the UCS buffer and is executed with CZCMD3 specified as its posting entry point. SETUR then exits to the caller with a return code for incomplete. If an error occurred when reading the SYSUCS record, the SYSUCS file is closed,

DCBSUR is set off, and SETUR returns to the caller with a return code for an invalid SYSUCS key or data set. VAM error return information may be found in the MSAM DCB.

If the SDAT and SYSURS values for the UCS key match, the UCS buffer does not have to be loaded. If the UCS strikeout character is to be used, it is tested for validity (ABEND results if it is invalid) and converted and saved in the DCB. The SYSURS folding code is saved in the SDAT, and SETUR proceeds to test for printer alignment (see below).

If the UCS feature is not in use, loading the UCS buffer is not necessary, so testing for print alignment occurs immediately.

If print alignment is necessary (unless an RJE task), an IORCB specifying CZCMD3 as its posting entry point is built to print 50 lines on the printer for purposes of alignment. A message is sent to the operator via WTO requesting him to align the printer, and SETUR returns an incomplete while operator response is awaited. CZCMD2 is specified as the asynchronous entry point.

If no alignment is necessary, or if the task is RJE (in which case alignment is not possible), processing is completed. The URS key from the setup parameter is stored in SDAT, DCBSUR is set off, any active interruption is deleted, (not necessary if the task is RJE), and control returns to the caller with the return code in register 15 set to "completed successfully" or "completed with unrecoverable I/O error."

DCBSUR on, DCBRCX not in the range 100 through 136: SETUR abnormally terminates when this invalid condition occurs.

DCBSUR on, DCBRCX = 100: The operator has not yet mounted the requested form on the card punch. Until he does, control is returned to the caller with a return code for incomplete.

DCBSUR on, DCBRCX = 104: The operator has mounted the specified form on the card punch as requested. The punch form number from the setup parameter is therefore moved into the corresponding field in the SDAT, and normal ending procedures (see above) are followed.

DCBSUR on, DCBRCX = 108: The operator has not yet mounted or set the SYSURS-specified form, carriage tape, chain/train or density on the printer as requested. Control returns to the user with a return code for incomplete.

DCBSUR on, DCBRCX = 112: The operator has mounted or set the requested form, carriage tape, chain/train, and density for the printer. The corresponding four SDAT fields are therefore set from the SYSURS-specified values, and processing continues as if the four SDAT fields were already correctly set (see above).

DCBSUR on, DCBRCX = 116: SETUR is in the process of loading the UCS buffer. If the buffer loading operation is not yet complete, control returns to the caller with a return code for incomplete. If the loading is complete, but the DEB indicates an error, DCBSUR is turned off and SETUR exits to its caller with a message to the operator and return codes for "unrecoverable error". If intervention is required, a message is sent to the operator with a return code indicating incomplete. If there is no error, an IORCB specifying CZCMD3 as its posting entry point is built and executed, to print the UCS buffer along with the UCS form of the verification message, and control returns to the caller with a return code for incomplete.

DCBSUR on, DCBRCX = 120: SETUR is in the process of printing the UCS buffer and verification message. If the printing is not yet complete, control returns to the caller with a return code for incomplete. If the printing is complete, but the DEB indicates an error, DCBSUR is turned off and SETUR exits to the caller with a message to the operator and return codes for "unrecoverable error". If intervention is required, the message is sent to the operator with return code indicating incomplete. If there is no error, a WTO is issued to write the EBCDIC form of the verification message on the operator's console, and to request that the operator verify that this verification message matches the one previously printed on the printer. CZCMD2 is specified as the asynchronous entry point, and SETUR returns to the caller with a return code for incomplete while operator response is awaited.

DCBSUR on, DCBRCX = 124: The operator has not yet verified that the print line appeared identical on the printer and the console. Control is returned to the caller with a return code for incomplete.

DCBSUR on, DCBRCX = 128: The operator has verified the UCS printing. The UCS key is stored in the SDAT and the universal-character-set bit is turned on in the DCB. Processing then continues as if the SDAT and SYSURS values for the UCS key matched (see above).

DCBSUR on, DCBRCX = 132: The alignment lines are being printed and the operator is checking printer alignment. If the DEB indicates an error, any active interruption is deleted; DCBSUR is turned off, and con-

trol returns to the caller with a message to the operator and return code for "unrecoverable error". If no error is indicated, control returns to the caller with a return code for incomplete.

DCBSUR on, DCBRCX = 136: The operator has successfully aligned his print form. If a print IORCB is still outstanding, control is returned to the caller with a return code for incomplete. If there is no outstanding IORCB, processing continues as if no alignment were necessary (see above).

DCBSUR on, DCBRCX = 140: A start I/O error retry has failed and DCBRCX is set to 140 before returning from the SETUR synchronous interruption routine entered at CZCMD3. At the next SETUR macro instruction, the SETUR routine, entered at its main entry point, will detect this condition, set a return code of 8 (unrecoverable I/O error) in register 15, and return.

DCBSUR off, DCBRCX not less than 100: The caller has turned off the DCB SETUR-in-progress bit in order to prematurely terminate the SETUR processing. In this case, the SDAT carriage tape, chain/train, folding option, UCS key, URS key, and density fields are zeroed if the device is a printer.

ASYNCHRONOUS INTERRUPT ENTRY AT CZCMD2: SETUR is given control at its asynchronous entry point, CZCMD2, when an operator response (an attention interruption caused by changing the state of a device from "not ready" to "ready") is received.

If neither printer alignment nor intervention-required retry is in progress, DCBRCX is incremented by 4. Otherwise, unless there is an outstanding IORCB or error indicated, a RESET will be issued and an IOCAL SVC executed. Then, unless printer alignment is in progress or the task is RJE, the interruption routine will be deleted. In all cases, control returns to the caller with a return code of 0 in register 15.

SYNCHRONOUS INTERRUPTION ENTRY AT CZCMD3: The synchronous entry point of SETUR is given control by the task monitor when the I/O activity associated with an IORCB terminates.

Errors occurring during the I/O activity will result in either no retry or limited retry, depending upon the type of error. Any final error is recorded in the DEB.

If a unit check or unit exception is indicated when alignment is in progress, but no intervention is required and no errors are indicated, DCBRCX is incremented by 4 to indicate completion of alignment, and the number of outstanding IORCBs is reduced to zero. Then, as in all cases, control returns to the task monitor with a return code of 0 in register 15.

In an RJE task, if a unit check with intervention required is detected, a return code of 20 will be placed in the DECB for reference by BULKIO. If, for an RJE task, a unit check is not detected but a unit exception or incorrect length is detected, set-up for retry after continuation is specified and return is made to the Task Monitor. (The remote operator must feed in a CONTINUE card for the job to continue; this will cause the next entry to be at the asynchronous entry point, CZCMD2.)

TAM PROCESSING

The following routine is used with Terminal Access Method (TAM) processing.

TAM Open Routine (CZCYA)

In continuing the open processing from Open Common, TAM Open is called to perform additional opening functions for terminals. This includes building control blocks and providing buffer areas for initiating communications with a terminal. TAM Open then returns to Open Common except when an abnormal end is required, in which case it goes to ABEND. (See Chart AK.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged.

Entry Point: CZCYA1 -- Entered by type-1 linkage.

Input: When this routine is entered, register 1 contains the address of the following parameter list:

Word 1 -- Address of the current DCB being opened.

Word 2 -- Address of the associated JFCB.

Data References: CHADCB, CHATDT, CHASDA, CHADEB, CHATOS.

Modules Called:
Write (CZCYM) -- TAM write.

Check (CZCRC) -- Check.

VMA (CZCGA) -- Get virtual storage.

LVPRV (CZCJL) -- Leave privilege mode.

WTO (CZABQ) -- Write to operator.

ABEND (CZACP) -- Abnormal task termination.

ADDEV (CEAAC) -- Add device to task device list.

RMDEV (CEAAD) -- Remove device from task device list.

Exits:
Normal -- Normal RETURN to calling routine.

Error -- ABEND macro instruction.

Operation: TAM Open initially saves the general registers, gets the TOS page and examines the number of DCBs that were opened for a given terminal.

- If this number is equal to 0, the terminal definition is checked.
- If this number is greater than 0 and less than or equal to 255, TAM Open continues the processing by updating the count of DCBs in the SDAT.
- If this number is greater than 255 (the maximum number of opened DCBs allowed for a terminal), TAM Open branches to ABEND.

The terminal is checked for definition and type when the first DCB is opened for this terminal.

- If the terminal is defined, TAM Open continues the processing by updating the count of DCBs in the SDAT.
- If the terminal is not defined this routine branches to ABEND.
- If the terminal is defined and is either a 2741 or a 1050, the system must determine which of the two terminal types is involved. This determination is made by issuing the LCD macro instruction.

TAM Open issues a GETMAIN macro instruction to obtain one page of virtual storage for the data extent block (DEB), and the terminal operational status table (TOS) which includes the I/O request control block (IORCB). Pointers are set up between these blocks and blocks that were created before TAM Open, and are illustrated in Figure 4. The purpose of these pointers between TAM Open allocated blocks and previously existing blocks is to facilitate their use by all TAM routines under normal and abnormal conditions.

DEB virtual storage, after being initially allocated during TAM Open, has the following pointers set and data moved:

(A) The DEB is pointed to by the DCB.

(B) The DEB points to the DCB and the SDAT.

(C) The DEB points to the SDAT terminal information. This SDAT terminal information pointer is moved from the JFCB.

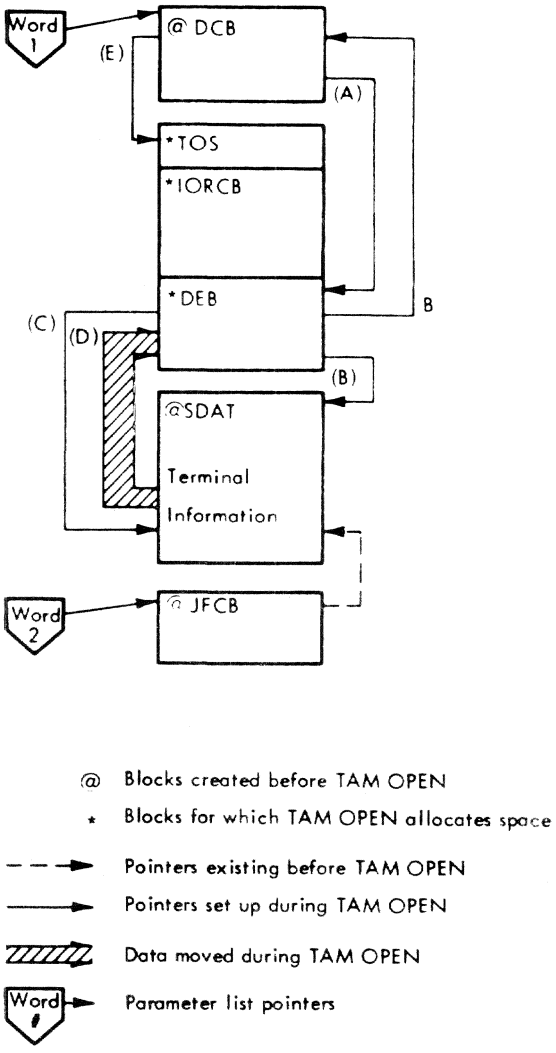


Figure 4. TAM Open: DEB and TOS Storage Allocation and Pointers

(D) The SDAT terminal information is loaded into the DEB. This terminal information is loaded into the DEB. This terminal information is used by the TAM Read/Write and TAM routines and includes the terminal type, control unit type, data adapter and model code of the terminal.

TOS virtual storage is allocated by TAM OPEN and is pointed to by the DCB. TAM Read/Write uses the TOS area to build the channel command words. It is also used by TAM Read/Write, TAM Posting, and TAM Open for communication of common information.

Processing ends with the terminal DCB open count in SDAT incremented by 1 for this current DCB and a return to Open Common.

IOR PROCESSING

The following routine is used with Input/Output Request (IOR) processing.

IOR Open Routine (CZCSC)

In continuing the open processing from Open Common, IOR Open is called to allocate storage for control blocks and complete the required fields in the DEB to allow the processing of the IOREQ macro instruction.

IOR Open then returns to Open Common except when an abnormal end is required, in which case it goes to ABEND. (See Chart AL.)

Attributes: Reentrant, resident in virtual storage, closed, nonrecursive, read-only, privileged.

Entry Point: CZCSC1 -- Entered with type-1 linkage.

Input: When this routine is entered, register 1 contains the address of the following parameter list:

Word 1 -- Address of DCB being opened.

Word 2 -- Address of associated JFCB.

Word 3 -- Address of workpage obtained by Open Common.

Data References: CHADCB, CHATDT, CHASDA, CHADEB

Modules Called:

VMA (CACGA) -- Get virtual storage.

CKCLS (CEAQ4) -- Check protection class.

Exits:

Normal -- Return to calling routine.

Error -- ABEND macro instruction.

Operation: IOR OPEN saves the general registers. Tests are then made to check that:

- The IOREQ facility is allowed on the device, by checking that the SDAIOR field in SDAT is equal to one.
- The DCB identification is valid, by checking that the DCBID identifier is equal to *%*%.
- The TDT indicates that the IOREQ facility is specified in the DD command (TDTDSV=RX). The DCB was previously checked in Open Common (DCBDSO=RX) which called for IOR Open. The data set organization requirements must be specified at data definition time only.
- The user-supplied NCP parameter, in the DCB macro instruction, is not greater than the maximum (DCBNCP=99). If the user did not set a value, or set a value of zero, a value of 1 is inserted.
- The volume is not public, as indicated by the volume public flag (TDTV1=0) not being set. IOREQ can only be used on private volumes.

An ABEND exit is taken if any of the above error conditions occur. Processing continues by caluclating the area needed for the DEB plus additional contiguous bytes for the IORCB. The length of the DEB is set to contain a common area plus extra storage for each NCP specified. This amount is rounded to a multiple of eight so that the contiguous bytes to build the IORCB originate on a doubleword boundary.

IOR Open then requests, with a GETMAIN macro instruction, one page of virtual storage for DEB and IORCB to be used by IOREQ and IOREQ Posting. This area is initially zeroed.

DEB virtual storage, after being initially allocated during IOR Open, has the following pointers set and data moved (refer to Figure 5):

(A) The DEB is pointed to by the DCB and JFCB.

(B) The DEB points to the DCB and JFCB.

(C) The DEB points to the IORCB.

(D) The DEB points to the SDAT address and device address information. These SDAT information pointers are contained in the JFCB.

(E) The JFCB data type code information is moved to the DEB.

Some other DEB fields listed below are also completed:

- The identification field (DEBID) is set to *(.
- The size field (DEBSIZ) is set to DEB area size.
- The IOREQ Posting VCON (DEBPSV) is set to IOREQ Posting entry point and the RCON (DEBPSR) is set to IOREQ Posting PSECT address.

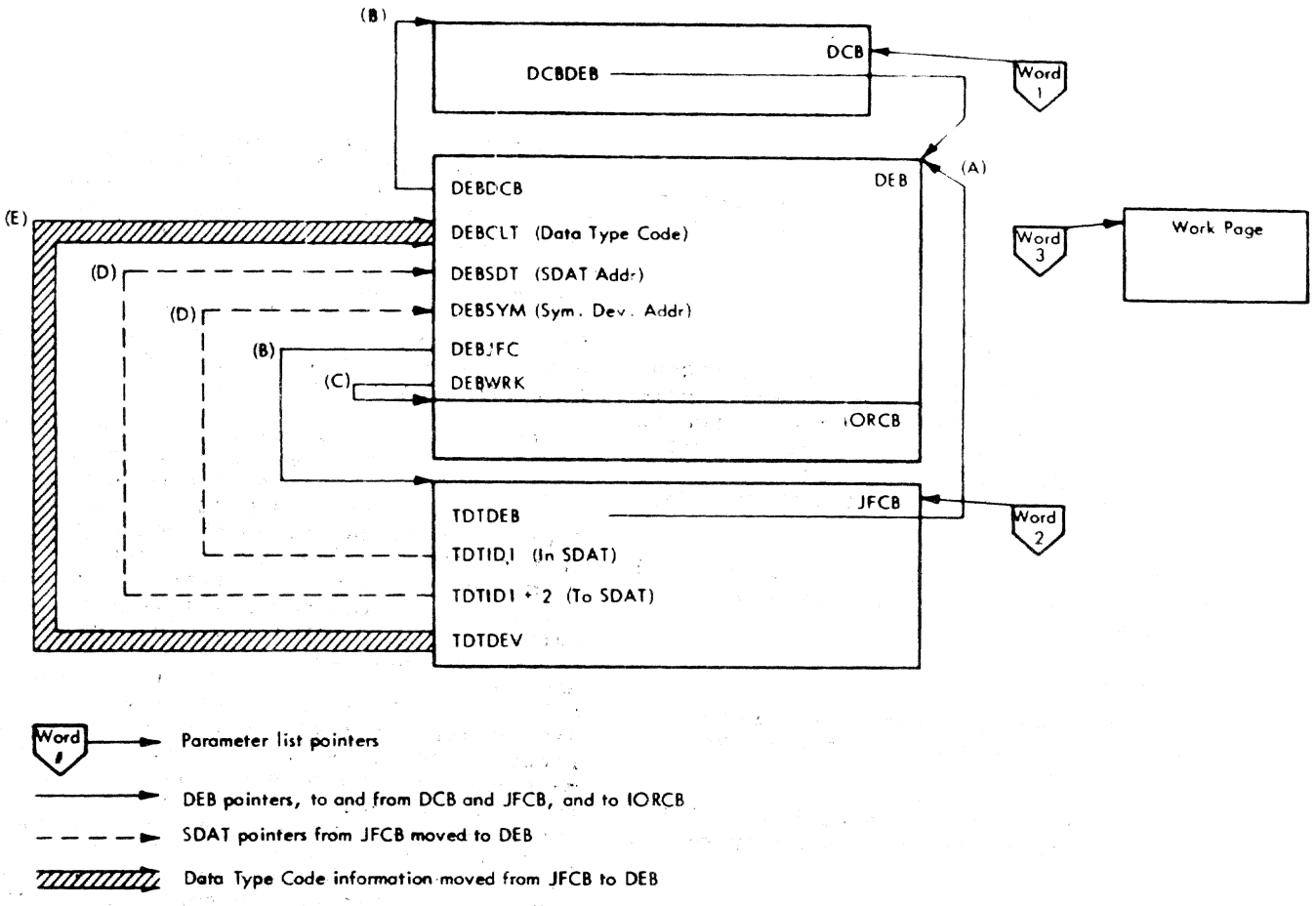


Figure 5. IOR OPEN: Basic Pointers and Data Moved from JFCB to DEB

- The number of channel programs (NCP) field in DEB (DEBNP) is set to the value in the DCB field (DCBNP).
- The DEBDVC field is set with a hex code indicating the device (magnetic tape, direct access, unit record).
- The DEBUNT field is set with a hex code indicating the unit type (2400, 2311, ...).
- The maximum number of IORCBs allowed field in the DEB (DEBIO) is set to the maximum in the SDAT field (SDAMRB).
- The DCB protection class field in the DEB (DEBCLS) is set to the appropriate value. It is as obtained as a result of an SVC, generated by a CKCLS macro instruction. A final check on this value within IOR Open assures that the user is privileged if the access to the volume is privileged.
- A pointer to the DECB field in the DEB (DEBDEL) is not completed during IOR Open but during IOREQ. However, the address of DEBDEL is stored in a DCB field (DCBDEC) during IOR Open.

A final check is made of both the DCB protection class and the type of access to the device. Then,

- If the DCB protection class is not privileged (DEBCLS+1) and the type of access to the device is privileged (TDTVPY=1), an ABEND exit occurs.
- If no DCB protection class can be determined, an ABEND exit occurs.
- If neither of the above criteria are met, a return to Open Common occurs.

SECTION 3: READ/WRITE

READ/WRITE PROCESSING

This section describes the BSAM Read and Write, MSAM Get, Put, Read and Write, TAM Read and Write, and IOREQ routines.

BSAM Read/Write Routine (CZCRA)

The Read/Write routine creates an IORCB which contains the appropriate channel commands to perform the I/O operation which is requested by the BSAM READ or WRITE macro instruction. The IORCB is passed to the I/O supervisor so the channel commands may be executed. (See Chart BA.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged, nonrecursive.

Entry Points:

CZCRAS -- Main entry point to the Read/Write routine; entered by type-1 or type-2 linkage.

CZCRDS -- Entry point for building an IORCB (Build subroutine); entered by type-1 linkage.

CZCRES -- Entry point for adding the channel program (Construct subroutine); entered by type-1 linkage.

Input: Register 1 contains the address of a DECB containing all information which was coded as parameters of a Read or Write instruction.

Data References: CHASDA, CHAIOR, CHADCB, CHADEB, CHADEC, CHAVPS.

Modules Called:
AWAIT (CEAP7) -- Await an interruption.

IOCAL (CEAA0) -- I/O call.

CKCLS (CEAQ4) -- Check protection class.

ASCII Translation and Conversion (CZCWA) -- When an ASCII Write request is encountered.

Exits:
Normal -- Return to the calling routine.

Error -- Abnormal termination of task by ABEND macro instruction.

Operation: The main entry point processing at CZCRAS is as follows:

During open processing, a queue of zeroed DECB pointers is established in the DEB for the data set. The main routine stores the address of the input DECB pointers in the reserved pointer area.

When this routine is entered, if the passed DECB is the same as the first DECB in the queue, it is the Check routine that has entered the Main routine to have a read or write request reissued.

If entry was not by Check, the new DECB is inserted into the queue if there is room. The "in use" and "intercepted" flags are set on in the new DECB. Should the "number of IORCBs allowed" count permit another IORCB to be given to the I/O supervisor, a search of the queue of DECBs is initiated to find the first request to be fulfilled. If that particular DECB is associated with an ASCII Write request that was intercepted by the I/O supervisor, a call to the ASCII Translation and Conversion routine (CZCWA) is made to put the record back in EBCDIC format. Then the main routine proceeds to have the IORCB built by the Build routine and filled in by the Construct routine. Successful construction of an IORCB will then permit the main routine to execute the IOCAL SVC which requests the I/O supervisor to perform the I/O operation.

The build entry point processing at CZCRDS is as follows: The Build routine generates a skeleton IORCB and stores several parameters in it.

Build returns to the Main routine which stores the address of the SAM Posting routine in the IORCB, and branches to the Construct routine to complete the IORCB by constructing a channel program in it.

The construct entry point processing at CZCRES is as follows: The Construct routine is passed a pointer to the skeleton IORCB which was generated by Build. The Construct routine must complete the IORCB so that it may be used by IOCAL. Familiarity with IOCAL as presented in System Programmer's Guide, GC28-2008, is required to understand the logic of Construct .

The first decision Construct must make is whether to use the IORCB as a buffer or to build a page list which points to the user's data pages. If the data exceeds 1800 bytes, the IORCB cannot be used as a buffer, and a page list must be constructed.

For 7-track tape the channel program which is generated consists of two CCWs. The first CCW is a Mode Set command which sets the density, data converter, and translator as specified in the tape option field in the DCB. The second CCW is the read or write command. For 9-track tape, just the one CCW for the read or write is required.

For an ASCII write request, ASCII Translation and Conversion is called to translate the data to be written from EBCDIC to ASCII and convert variable-length records from V to D format (standard IBM to American National Standard). ASCII output may occur only to 9-track tape.

For direct access, the channel program is more complex. The generated channel program depends on whether the requested operation is a read or write, whether track overflow is specified, and what the record format is.

The channel program for a DA read is generated as follows. A full DA device address is in the form bbCCHHR, where bb is two bytes which indicate bin number (data cell), CC is two bytes which indicate cylinder number, HH is two bytes which indicate track (or head) number, and R is the record number on the track. The first CCWs are two Seek commands to the next I/O address (bbCCHH) which is found in the DEB. The first Seek command is software chained because a seek takes a relatively long time to execute (arm movement may be necessary), and hardware chaining of commands makes the channel unavailable to other tasks. Therefore, software chaining will cause the two seeks to be executed without locking other users out of the channel. The second seek command is command chained to the rest of the CCWs in the channel program, so execution of the second seek gives the task exclusive control of the channel until the execution of the channel program is concluded. The next CCW is a search for the identifier CCHHR, which will position the DA device to the key field of the record. A TIC command follows the search to cause the channel program to loop until the record is found. The Read command (read key and data) is next in the channel program and it is followed by a NOP command.

When reading format-U or V records, it is not known where the next record to be read is. This is because the number of bytes in the current record is not known. If the end of the current record cannot be calculated, the beginning of the next record is not known. So when reading format-U records, the search command is directed to the last I/O address. Then three CCWs are generated: read-no transmit, read, and NOP. The READ-no transmit causes the last record to be passed by the reading head, and the read key and data causes the correct record to be read.

The channel program for a DA write is generated as follows. Using the last write address from the DEB, the first four channel commands are generated: seek, seek, search, TIC. The reason for using the last I/O address is that it is known where the last record was written, but if format-U records are being written, it is not known where the next record should go. Then a read-no transmit, followed by a write-count, key, data is generated. The Read-no transmit brings the head past the last record which was written and then the WRITE puts the new record on the volume sequentially following the previous record. If track overflow is specified and the record to be written is larger than the room remaining on the current track, the record is begun on the current track and continues on the next sequential track. However, no record can be split between cylinders on a DA volume.

As in the channel program for a DA read, the write channel program is ended by a NOP.

The Construct routine uses closed subroutines to help in the completion of the IORCB. They are:

ENTCCW -- Enters a CCW into the IORCB.

SCHCCW -- Generates SEEK, SEARCH, and TIC CCWs, and puts them in the IORCB.

SETPAG -- Sets up a page list in the IORCB.

NXTIO -- Computes the MBBCCHHR of the next record to be written on a DA volume.

Intercepting a Read or Write Request: The DECB which is passed to the SAM Read/Write routine contains an intercept flag set by SAM Read/Write when the I/O request is not actually initiated. (IOCAL is not given the IORCB built for the DECB).

There are several reasons why the I/O request of a particular DECB may not be initiated. For example, if a read request is given and the current volume was used up on the preceding read, the DECB is set as intercepted and the EOV request flag in the DECB is set on. If a write request is given and there is no more room in the currently allocated extents, the same flags are set. If the DECBs request I/O on the same devices and the first one encounters a hardware error condition, the intercept flags in the remaining DECBs are set on so the I/O will not be initiated unless the hardware error is cleared up. If this were not done, each DECB I/O request could run

into the same hardware problem and a permanent error would be found, not once, but many times.

When a DECB is operated upon by the Check routine, the intercepted flag is tested. Should there be no serious error conditions posted in the DECB, the Check routine, in addition to performing other services, will reinitiate the I/O request by passing the DECB back to the Read/Write routine. This I/O request is given immediate attention by Read/Write because it is the first entry in the queue.

DOMSAM Routine (CZCME)

DOMSAM blocks and deblocks logical records into system buffers when GETs or PUTs are issued, invokes MSAM READ/WRITE to build and execute the IORCB necessary for I/O, and provides the user with a return code describing the outcome of his GET or PUT request. The routine runs in the same privilege as its caller. (See Chart BB.)

Attributes: Read-only, public, reentrant, nonrecursive, assumes privilege of caller.

Entry Points:
CZCME1 -- Entered upon issuance of a GET macro instruction which generates type-1 linkage.

CZCME2 -- Entered upon issuance of a PUT macro instruction which generates type-1 linkage.

Input:
Register 0 -- Address of user-specified area for move mode GET or PUT.

Register 1 -- Address of the DCB.

Data References: CHADCB, CHADEB, CHADEC, CHADBP, CHAISA.

Modules Called: MSAM Read/Write (CZCMF) -- To read or write records.

Exits:
Normal -- Following a GET, register 15 contains one of the following return codes:

'00' Normal completion. Register 1 points to record obtained in buffer if locate mode, user area if move mode.

'04' Request incomplete.

'08' Unrecoverable I/O error. Register 1 points to (failing) record in buffer if locate mode, in user area if move mode register 0 points to user DECB.

'0C' End of data set reading. (No record obtained). Register 0 points to user DECB.

'10' Control card sensed reading. Register 1 points to control card in buffer if locate mode, user area if move mode. Register 0 points to user DECB

'14' Intervention is required in an RJE task because the line is disconnected. (No record is obtained.)

Normal -- Following a PUT, register 15 contains one of the following return codes:

'00' Normal completion. Register 1 points to next available location in buffer if locate mode, to user area if move mode.

'04' Request incomplete.

'08' Unrecoverable I/O error. Register 1 points to buffer of record that failed to be written. Register 0 points to user DECB.

'0C' Intervention is required in an RJE task because the line is disconnected. (No record is written)

Error -- ABEND macro instruction is used for abnormal end termination.

Operation: DOMSAM has no PSECT. It uses the first 19 fullwords of the DEB work page as its standard register save area, obtains adcons from the DEB page (CHADBP), and maintains switches and other variable information in the MSAM portion of the DCB. Its work areas are the DEB work page, which contains DECBs, and the buffer pages.

Upon entry to DOMSAM, a transfer pointer is set to indicate whether a GET or a PUT has been issued. Checks are then performed to be certain that the DCB and the DEB are valid, and that the DCB has been opened. If any one of the three conditions is not met, execution is terminated by an ABEND macro instruction. If all conditions are met, processing is categorized as GET or PUT.

GET Processing: If the GET macro instruction is the first issued on the data group, or if the previous GET emptied a buffer, the end-of-buffer switch in the DCB will be on. In this case, MSAM Read/Write must be invoked to either prime all the buffers or refill the buffer just processed. If priming is to be done, no IORCBs may be outstanding or the return code will be set to indicate incomplete, and a return will be made to the caller.

For each buffer to be filled, the current DECB is marked in use (read/write requested) and initialized, and MSAM Read/Write is invoked. Upon return from Read/Write, the current DECB is checked to see if it is the last in the list, and if so,

its pointer is reset to point to the first DECB. If not, the DECB pointer is incremented to point to the next DECB.

A check is then made to see if all the buffers are to be primed, and if so, the next buffer is filled by setting up the DECB and invoking MSAM Read/Write as before, until the last DECB has been processed. At that point, the current DECB address is reset to point to the first DECB.

The DCB pointer to the current buffer page is set to the address of the buffer associated with the current DECB. Then the DECB is tested for completion, and if the I/O is not yet complete, control is returned to the caller with a return code indicating that the GET has not been completed and must be reissued at a later time. In an RJE task, if the line to the remote device is not connected, the return code will indicate intervention is required.

If the DECB is posted complete and indicates normal completion with neither a unit check (indicating that a control card has been read) nor a unit exception (indicating end of data set), the normal completion code of zero is set into the DCB for eventual use as a return code. The pointer to the current logical record within the buffer is then set immediately beyond the 32 control bytes in the buffer, and the end-of-buffer address is computed by adding to that address the product of the logical record length and number of records that can fit in the buffer. Following a transmission from a remote reader (RJE task), this end-of-buffer address is adjusted based on whether an odd or even number of logical records were read into the buffer.

If the DECB is posted complete without errors, but unit check or unit exception is indicated, or if the DECB is posted complete with errors, the appropriate return code is set into the DCB, and a copy of the current DECB is moved to the user's DECB area. In this case, the end-of-buffer address is computed as the byte immediately following the last normal input record by adding the displacement-to-error field of the DECB's modified CSW to the beginning-of-buffer address. This displacement must be decremented by 84 if, in an RJE task, only one logical record was read in during the last transmission. The current logical record address is computed as 32 bytes beyond the beginning of the buffer.

Following this, or if no buffer priming or refilling was necessary to begin with, a check is made to determine if the current logical record is valid by comparing the record address with the end-of-buffer address. If it is lower, normal steps will be taken to get the record for the user. Otherwise, one of the unusual conditions (end-of-data-set, control card, or record with error) exists, and the return code in the DCB is set into register 15 for the user. The end-of-buffer and buffer-priming switches are set, the count of logical records within the buffer is set to zero, and the pointer to the current DECB is reset to point to the first DECB in the list so that the next GET issued will reprime all the buffers. If the unusual condition is an end-of-data set, there is no record to be obtained for the user, and control is returned immediately. However, if there is a record beyond the buffer end address (either a control card or a record with an error), it is returned to the user as described below.

When the current logical record is valid, it must be returned to the user. If the GET is in locate mode, the current record address is set into register 1; if the GET is in move mode, the record is moved to the user-specified area whose address was supplied in register 0. The current record address is then incremented by the logical record length to point to the next record, and the count of logical records already processed within the buffer is incremented by one. If the record address is no longer less than the end-of-buffer address, or if the count of logical records is 100, the current buffer is completely processed. In that case, the end-of-buffer switch in the DCB is set on to indicate that end-of-buffer processing is necessary before the next GET can be completed, and the count of logical records is reinitialized to zero. Whether or not this GET emptied a buffer, the return code is set to zero, signifying normal completion of the GET, and control is returned to the caller.

If, upon entry to DOMSAM, the end-of-buffer switch is on, and if a FINISH macro instruction was previously issued, edits and initialization largely of the type performed by MSAM Open are required before MSAM Read/Write may be invoked to reprime all the buffers. If any permanent errors have occurred, or if any IORCBs remain outstanding, the task is abnormally terminated; if not, a flag is set in the DCB to indicate that this is the first GET issued on the data group, the buffer-priming switch is set on, and the FINISH-just-issued flag is turned off.

The record format is checked to be certain it is not variable. If variable, the task is abnormally terminated since variable format records are not supported for the card reader. If not variable, the maximum allowable record length is calcu-

lated, and compared with the value specified by the user. If the user-specified value is not greater than zero and less than or equal to the computed maximum, the task is abnormally terminated. If the record length is acceptable, the count of logical records within the buffer is set to zero, and the maximum number of logical records per buffer is computed and stored in each DECB. The current DECB pointer is set to point to the first DECB in the list, the acknowledgement (ACK) switch is initialized for RJE, and the routine proceeds with buffer priming as described above.

PUT Processing: Initially, for both local and RJE jobs, if the PUT before the current one was not in locate mode, and if no end-of-buffer processing is required, the following processing occurs.

The length of the record to be PUT is obtained either from the DCB, or, if the record is variable format and the PUT is in move mode, from the length control bytes in front of the record. If variable, the length is checked for validity -- at least equal to the number of control bytes (0, 1, 4, or 5), but no greater than the maximum for records of its type. An invalid length causes abnormal termination of the task.

For local devices, a record of the obtained length will not fit in the space remaining in the buffer, end-of-buffer processing must be done before the current record can be processed, and control is transferred to a section for invoking MSAM Read/Write. If the maximum size record will fit within the buffer, and if the PUT is in locate mode, the current record address is set into register 1, the return code is set to zero to indicate normal completion and control is returned to the user. If the PUT is in move mode, the record in the user-specified area whose address was supplied in register 0 is moved to the current buffer location.

For RJE (which operates only in locate mode), buffering is handled differently than for local devices. Logical records are built into transmission control blocks (TCBs) aligned on halfword boundaries within the page buffer. The page buffer and DECB work page both contain appropriate pointers and flags (current TCB pointer, final TCB flag, etc.). The address pointer returned to the caller does not point to an area in the page buffer itself, but rather to a 144-byte area within the DECB work page. DOMSAM then processes the record from this location, moving it to the current TCB in the page buffer after it has been processed. On normal completion, the address of the record buffer in the DECB work page is returned to the user in register 1.

Additional processing depends on whether or not the TCB is filled, whether it is the final TCB in the page buffer, and, if not, whether there is sufficient space in the buffer for another TCB of maximum (404 bytes) or minimum (278 bytes) length. Appropriate flags are set; if an end-of-buffer condition is reached, MSAM Read/Write is called. If the RJE device has the multiple record feature (MRF), seven records may be placed in a TCB; otherwise, two. If necessary, the 'Previous Put in Locate Mode' flag is set on before returning to the user.

If for both local and RJE, the previous PUT was in locate mode, the record subsequently built in the buffer requires checking. If variable format records are being used, the length is checked for validity as above and also for being no greater than predicted. If the DCB indicates that this is a form-sensitive file for a local or remote printer, the control character is tested. A machine code control character specifying "skip to Channel 1 after print" will trigger the ending of the current buffer after this record. A FORTRAN (ASA) control character specifying "skip to channel 1 before print" will result in the new page switch being set in the associated DECB as this is the first record in the buffer, or will trigger processing to end the buffer in front of this record if it is not.

For RJE tasks, RJE transmission control characters are inserted into each record, and a dummy record is moved to the TCB where this is the first PUT after OPEN or FINISH, or if this is the first record in the buffer and it has FORTRAN (ASA) control characters. Trailing blanks are suppressed; error characters are translated from the record; if tabbing is required, tabs are inserted in the record; and the FORTRAN (ASA) or machine control character is translated to a 2780 control character. The processed record is then moved from the 144-byte record buffer in the work page to the current TCB in the page buffer and the TCB length is adjusted. If FINISH is in progress, an end-of-transmission TCB is added to the buffer.

In all cases, once the logical record is satisfactorily placed in the buffer, the current record address is incremented by the record length to point to the next available space in the buffer. If variable length records are being processed, the total block length is also incremeented by the record length. The count of records within the block is incremented by one, and if it has reached the output buffer maximum of 200, the end-of-buffer switch in the DCB is set on so that the next PUT processed

will cause end-of-buffer processing to occur.

Then, if processing of a move-mode PUT has just been completed, control is returned to the user with a return code of zero indicating normal completion. If, however, the previous processing has completed checking of a previous locate-mode PUT, control is transferred back to the beginning of the PUT routine, where the end-of-buffer switch is tested before processing the current PUT.

When a buffer must be ended and written out before the next record can be processed, the appropriate switches are set, and control is transferred to a section for invoking MSAM Read/Write. This section will check to see if Read/Write has already been invoked, and if so, will proceed to test the appropriate DECB for completion. If complete, control is returned immediately to the caller (MSAM FINISH).

If Read/Write has not yet been invoked, and if a FINISH macro instruction was not issued just prior to this PUT, the current record count is set into the DECB and then reinitialized to zero. The DECB is marked read/write requested, and MSAM Read/Write is given control. Upon return from Read/Write, the pointer to the current DECB is updated to point to the next DECB, unless the current DECB is the last in the list, in which case the pointer is reset to point to the first DECB in the list. The DCB pointer to the current buffer page is set to the buffer address associated with the current DECB, and the pointer to the DECB to be tested for completion is set. In an RJE task, TCB flags and pointers are initialized in the buffer and DECB work pages.

Once MSAM Read/Write has been invoked, the DECB is tested for completion. If it is not marked complete, the next record may not be processed, and control is returned to the user with a return code indicating incomplete. Where incomplete because an RJE line is disconnected, before returning to the caller, a return code is set to indicate RJE intervention is required.

If the DECB is marked complete, a check is made for an unrecoverable I/O error, and if one is found, a test is made to determine if the error was non-permanent and already returned to the user. In that case, provided no IORCBs are still outstanding (in which case a code of incomplete will be returned to the caller), MSAM Read/Write will be called to output the records suppressed by the error, and the DECB will be tested again for completion. If the error was not already returned to the user, the DECB associated with the error is located and moved to the user's DECB area. The address of the failing record is computed, and control is returned to the user with a return code indicating unrecoverable I/O error. If the proper DECB could not be found, it is assumed that error recovery is still in progress, and control is returned to the user with a return code indicating incomplete.

If no I/O error occurred and the finish-in-progress flag is on, control returns to the Finish routine. Otherwise, if no I/O error occurred, a check is made to determine if the new buffer begins a new form-sensitive print page. If so, and if FORTRAN (ASA) control characters are being used, the record which should start the new page is still trailing after the last record in the buffer just written and must be moved to the beginning of the current buffer. If the new buffer does not begin a new form-sensitive print page or FORTRAN (ASA) control characters are not used, the first available location in the buffer is set beyond the 32 control bytes. If the records are variable format, this address points to the block control bytes (LL) which are then initialized to four, and the first available location address is incremented by four. For RJE, this address pointer is further incremented by 12 to allow space for the printer selection sequence and various bisynchronous characters. The address of the first available location is saved in the DCB as the current record address, and control is transferred back to the beginning of the Put routine to process the current PUT.

If, upon entry to the Put routine, a FINISH macro instruction has just been issued and end-of-buffer processing is indicated, the following edits and initialization, largely of the type performed by MSAM Open, are required before the Put can be processed.

If any permanent I/O errors have occurred, or if any IORCBs remain outstanding, the task is abnormally terminated. If none, a switch is set in the DCB to indicate that this is the first PUT on this data group, and the various processing switches in the DCB are reset. All DECBs are reinitialized, the pointer to the current DECB is set to point to the first DECB in the list, and the current buffer page address is set from the current DECB. The record format is checked to be certain it is either fixed or variable, and if it is neither, the task is abnormally terminated. The current record address is computed as described above. In an RJE task, TCB pointers and flags are reinitialized and maximum and minimum TCB size is set. Abnormal termination occurs if the device is neither a card punch nor a printer, or

if the logical record length specified in the DCB is invalid. Otherwise, conrol is transferred to the beginning of the PUT routine to process the first record of the new data group.

MSAM Read/Write Routine (CZCMF)

The MSAM Read/Write routine builds an IORCB, which contains a channel program (CCWs) to process an entire buffer page of records. It invokes IOS by the IOCAL SVC to perform the actual input or output commands. (See Chart BC.)

Attributes: Privileged, read-only, public, reentrant, nonrecursive.

Entry Point: CZCMF1 -- Entry from DOMSAM (GET or PUT) by type-1 or type-2 linkage.

Input: Register 1 contains the address of the DECB.

Data References: CHADEC, CHADCB, CHADEB, CHAIOR, CHAISA.

Modules Called:
DIR (CZCJD) -- Delete interruption routine.

IOCAL (CEAA0) -- Initial supervisor processing of an IORCB.

SYSER (CEAIS) -- System error processor.

Reset (CEAAH) -- Reset device suppression flag routine.

Exits:
Normal -- No return codes used.

Error -- Link to ABEND with condition code 1 and appropriate error message.

Operation: When Read/Write is entered, the DCB address, the DEB address, and the address of the buffer page are obtained from the DECB. The location where the IORCB is to be built is obtained from the first system control word of the buffer page. Since this buffer page may be in Class A (user read-write) virtual storage, a series of checks are made to verify that the IORCB address is still valid and has not been changed by the user. An incorrect IORCB address causes an ABEND.

If the DECB passed to Read/Write indicates complete with error rather than Read/Write request, any outstanding asynchronous routine is deleted, the IORCB abnormally terminated is reissued immediately at a point beyond the command that failed, and any subsequent IORCBs whose DECBs are marked intercepted are reissued. Control then returns to the caller.

If the DECB passed to Read/Write indicates a READ/WRITE request, the IORCB is built and executed. The location of the buffer page is placed in the single page list entry. All CCWs created will reference this page list for the address of the buffer page. Pointers, counters, and the displacement are initialized. Each CCW contains a displacemnt field which corresponds to the displacement, in its buffer page, of the data to which this CCW refers. If the CCW does not reference data (for example, skip immediate or NOP), the displacement is the same as for the adjacent CCW. The device type is tested, and processing diverges for local or remote devices, and for input or output.

For local card readers, the data mode (EBCDIC or column binary) the stacker bin, and the record length are determined from the DCB. Two command-chained CCWs are generated for each record to be read; a read CCW followed by a feed and select stacker CCW.

After the CCWs have been constructed for a buffer full of records (count is given in the DECB), a NOP without command chaining terminates the CCW list. The length of the CCW list and the total length of the IORCB are computed and set in the fixed area of the IORCB.

If this is the first record of a data group, or on initial priming after an error, the device is reset for I/O and exceptional condition flags are cleared. (If not the first record, and an exceptional condition has been detected on the device, the DECB is marked "posting reissue" if error recovery is in progress, or, if not in progress, it is marked "intercepted", and control returns to the caller.)

Task interruptions are inhibited while the count of IORCBs outstanding is incremented by 1. Then this IORCB is issued to IOS by executing its IOCAL SVC. Control is returned to the calling program.

For remote card readers, an IORCB is built consisting of alternating read and write CCWs. Each read CCW brings in a 168-byte transmission containing two card records. With a remote card reader containing the multiple record feature, each read CCW will bring in four card records. Following each read CCW a two-byte write CCW is built in the IORCB to transmit to the device acknowledgement of the previous read. The write CCW transmits bisynchronous control characters (ACK0 or ACK1) which must alternate for successful transmission. For purposes of error recovery, the first CCW built in the IORCB will be a one-byte write containing a negative acknowledgement

character (NAK). This will be followed by a NOP command, then an initial write CCW with an acknowledgement character. The start CCW will follow and be the first of the alternating read and write CCWs built in the IORCB.

A page buffer in an RJE GET operation may contain up to 24 transmissions (48 card records). When the maximum number of CCWs have been reached (or there are no more records), a NOP command will terminate the command chain. The lengths of the CCW list and of the IORCB are computed and entered in the fixed area of the IORCB.

If no errors are outstanding, acknowledgement responses for the write CCWs are synchronized, task interruptions are permitted if necessary, and the IORCB is executed via the IOCAL SVC. Control is returned to the caller.

For card punches and local or remote printers both FORTRAN (ASA) and M control characters are supported. The record length for fixed length records is determined from the DCB. Variable length records specify their own length. The length field of a variable length record is never printed or punched nor is any control character printed or punched. Control bytes are examined in the buffer.

If the device is a card punch, one punch, feed, and select stacker CCW will be generated for each record to be punched. If M control characters are being used the control character itself is used as the command code. It must be a write command, otherwise ABEND terminates the task. If FORTRAN (ASA) control characters are being used, the command code is determined by combining the DCB mode specification with the appropriate extended ASA stacker specification. If no control characters are in use, the mode and stacker specifications are obtained from the DCB and combined to create the command code. If COMBINE is specified, the stacker must be RP3, otherwise ABEND terminates the task.

If the device is a local printer and if universal character set (UCS) printing is to be done a reset block data check command is the first command of a data group. Preceding the first print command of a data group is a skip immediate to channel 1 CCW. If neither FORTRAN nor M control characters are in use, each CCW created to print a line will be a write and space after writing. The number of lines to space is determined from the DCB. If M control characters are in use each control character will be used as the command code in the one CCW generated for each record. It must be a write command, otherwise ABEND will terminate the task. Since FORTRAN control characters specify skipping or spacing before printing as opposed to the command codes which specify only skipping and spacing alone, or skipping or spacing after printing, the control action is associated with the previous write, or requested alone if there is no previous write. Thus, if FORTRAN control characters are in use, the first record will cause two CCWs to be constructed; one control CCW to skip or space before printing and one write CCW. Each subsequent record will cause the generation of one CCW, whose write command code is also determined by the next FORTRAN control character. The CCW for the last record to be printed is a print and no space command. When the last punch or print CCW has been built, the list is terminated with a NOP, and the line of processing for the card reader is rejoined.

If the device is a remote printer, the channel program will consist of alternating write and read CCWs. Each write CCW will transmit a transmission control block (TCB) of data containing up to 7 records if the device has the multiple record feature and 2 records if it does not. Each read CCW will read in bisynchronous acknowledgement bytes for determining successful transmission (alternating ACK0 and ACK1). If this is the first PUT after an OPEN or FINISH macro instruction, or if the IORCB is being reissued for purposes of error recovery, the initial CCWs will consist of writes and reads of bisynchronous bytes to ascertain the state of the device and get the acknowledgements (ACK0 and ACK1) in synchronization and a selection record to select the printer. In addition, if the device has the tabbing feature, CCWs will be added to write the tabbing record, which will set the tabs on the printer. (The 144-byte tabbing record is built by Read/Write in the IORCB preceding the page list pointer; it is built only once for each IORCB.) The write/read CCW pairs will follow until an end-of-transmission (EOT) TCB is recognized. Then a write CCW is built for the EOT TCB and a NOP is added to terminate command chaining.

TAM Read/Write Routine (CZCYM)

During the execution of a system program, where the programmer had originally requested a read or write from or to a terminal, a call is generated that links to TAM Read/Write.

TAM Read/Write functions are accomplished by using the control blocks and buffer area allocated during TAM Open, and by using the tables internal to TAM Read/Write. The terminal-computer communication is accomplished through a buffer area under control of the channel command words (CCWs), located in the IORCB.

TAM Read/Write also issues control functions (orders) to the transmission control unit (2702). (See Chart BD.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged.

Entry Points:
CZCYM1 -- Type-1 linkage. Entered from systems routine.

CZCYM2 -- Type-1 linkage. Entered from TAM Posting.

Input: Register 1 contains the address of the DECB.

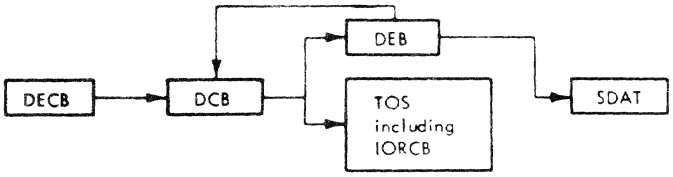
Data Reference: CHADEC, CHADCB, CHADEB, CHAIOR, CHATOS, CHASDA.

Modules Called:
ABEND (CZACP) -- Abnormal task termination.
CKCLS (CEAQ4) -- Check protection class.
IOCAL (CEAA0) -- I/O call.
TSEND (CEAP9) -- Time slice end.

Exits:
Normal -- Return to calling routine.

Error -- ABEND macro instruction.

Operation: TAM Read/Write initially saves the general registers. The general registers are then loaded with pointers to referenced control blocks, starting with the data event control block (DECB):



The DECB pointer to the DCB is included as one of the Read/Write macro instruction operands. The rest of the pointers are described in TAM Open.

An initial test is made to see if a nonprivileged user has called TAM Read/Write. An improper linkage will immediately ABEND the task.

Any time the command system is used, it causes TAM Read/Write to be invoked by way of the Gate routine. Nonprivileged programs should only use those GATE macro instructions which link to the Gate routine. The system program Gate routine then links to TAM Read/Write. A nonprivileged program attempting to directly call TAM Read/Write is an abnormal condition that causes a branch to ABEND. Two areas of error testing are then made to assure that system parameters are properly set. If either area is not satisified, a return to the user occurs. If both areas are correct, TAM Read/Write continues.

The first area tested assures that in the SDAT the number of active IORCBs is zero. No busy DECB exists for this terminal. (A busy DECB occurs after a TAM READ/WRITE is issued and ends during the corresponding TAM Posting.) If the number of active IORCBs is zero, the IORCB active count field in the TAM Read/Write PSECT for this terminal is set to 1 (for this TAM Read/Write entry) and the second area is tested. If the number of active IORCBs is not zero, then the in-use bit in the flag field of the DECB is set and a return is made to the user.

The second area tested assures that in the DECB, the type option code specified is within the range of all defined types for all terminal types, and that the type option code specified is valid for the actual terminal type.

If these conditions are met, data from referenced areas, including DEB information, are merged in TOS (terminal access operational status table) to reduce paging time later. The type option is then decoded.

If either of these conditions is not met, no further testing occurs. Instead, an error code is moved into the ECB of the DECB, the user error flag is set in the DECB and the SYNAD request is set in the DECB. A return to the user occurs.

The type option code in the DECB is then decoded to see if it is a control function (order). If it is a control function:

- The corresponding CCW generator generates one appropriate CCW in the TOS build area.
- The CCW is moved from the TOS build area into the IORCB CCW list area.
- The IORCB fixed area is completed.
- An IOCAL is issued, which links to IOS for this IORCB to be executed. When the supervisor returns control, TAM Read/Write restores the registers and returns to its user.

Accessing the Channel Program Generator:
If this is not a control function, TAM Read/Write begins processing to build a CCW list appropriate to the option requested for the particular terminal.

TAM Read/Write begins processing for this requested terminal computer I/O operation by first finding the proper channel program generator (CPG). To accomplish this a program search of internal tables within TAM Read/Write (illustrated in Figure 6) is required. These tables are:

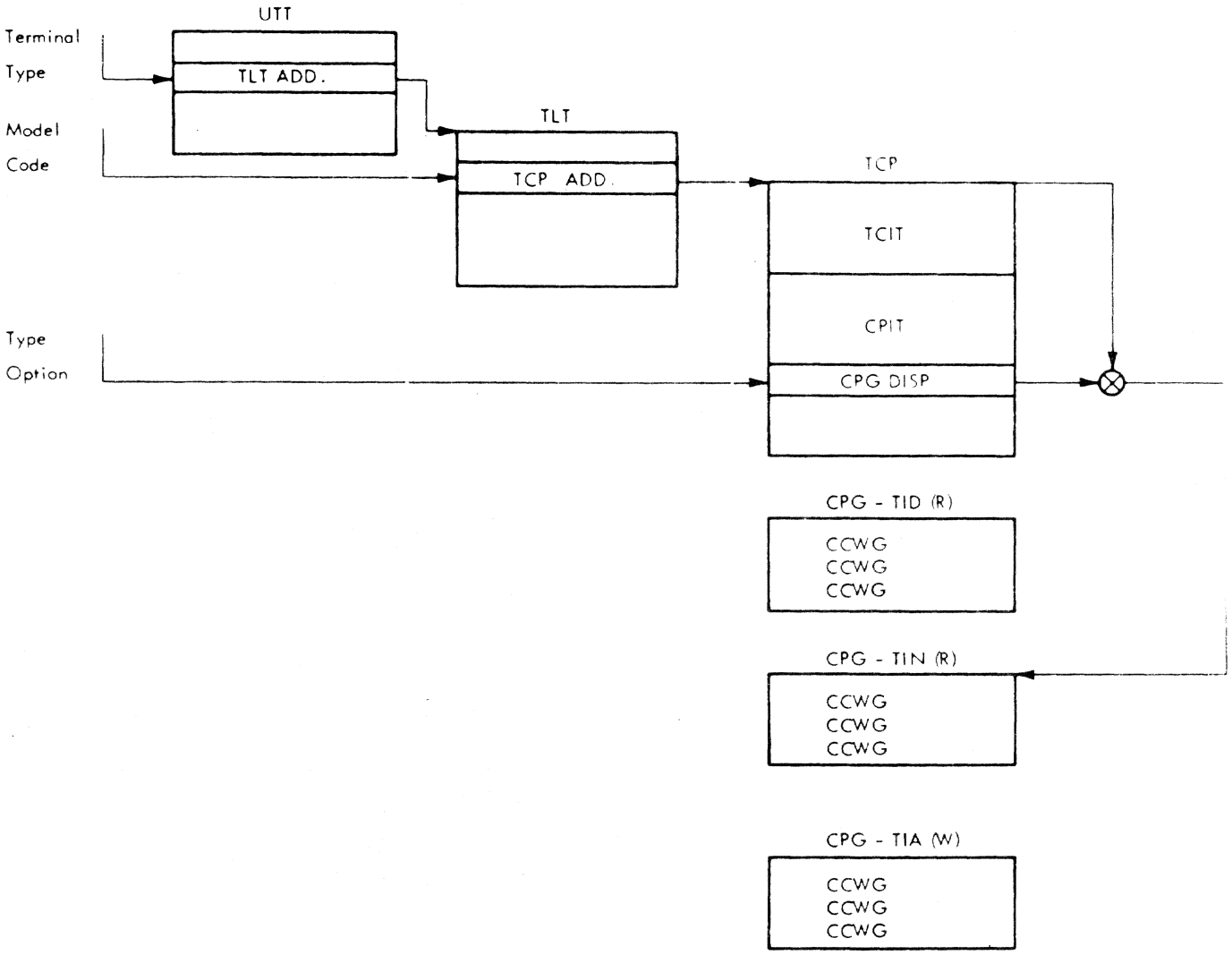


Figure 6. TAM Read/Write: CPG Location Sequence

- The unit type table (UTT)
- A terminal library table (TLT)
- A terminal channel program (TCP)

The procedure to find this CPG in TAM Read/Write requires as input:

- Terminal information loaded in the DEB from the SDAT during TAM Open containing terminal type and model code (refer to Table 8).
- Type option code specified in the DECB (refer to Table 9).

The sequence (refer to Figure 6) in finding this CPG is as follows:

1. The code for the mechanism for joining the terminal to the system (that is, a 2702 control unit connection or a direct multiplex connection) determines the displacement in the UTT to obtain the address of the TLT (Table 10).

2. The model code (that is, 1050 or a 2741) determines where in the TLT (refer to Table 11) the address of the TCP can be located. This address is then examined.

   a. If the entry in the TCP is zero, it indicates that the library does not exist.

   b. If the TCP entry is an address, TAM Read/Write then calculates a displacement from the TCP, which is used to get to the appropriate CCWG.

Table 8. TAM Read/Write: Terminal Information from SDAT

Model Code	Line Type	Terminal Type		SAD Order
		High-Order	Low-Order	
Byte 1	Byte 2	Byte 3	Byte 3	Byte 4
Byte 1 -- represents a model code for terminal units (1050, 2741, 35, 1052)				
Byte 2 -- represents a line type for connection (dial line or dedicated line)				
High-Order Byte 3 -- represents a terminal control unit type (IBM terminal control unit; type I, type II, telegraph terminal control unit; type I, type II, World Trade terminal control unit)				
Low-Order Byte 3 -- represents a terminal channel connection type (2702, Multiplexer)				
Byte 4 -- represents a SAD order (0, 1, 2, 3)				

3. In the TCP (Table 12), TAM Read/Write locates the proper CPG from the type option code. The TCP is divided into three main fields which are:

   - Terminal control information table (TCIT)
   - Channel program index table (CPIT)
   - Channel program generator (CPG)

The terminal control information table (TCIT) in the TCP contains data on addressing, polling, EOL sequence characters, etc., (Table 13).

The channel program index table (CPIT) (refer to Table 12) determines from the Type option code (stripped of the repeat bit if any) where the corresponding CPG displacement is.

The CPG is obtained from the displacement in the CPIT. This displacement value is added to the TCP address to generate the address of the CPG (Figure 6).

Makeup and processing of the Channel Program Generator: A CPG contains channel command word generators (CCWG) (refer to Table 14).

A breakdown of each of the CCWG sections is shown in Table 15.

To process the CPG, each sequence cycle begins by sequentially testing the CCWG buffer allocation flag bits shown in Table 16. For each flag that is on, TAM Read/Write branches to the indicated associated routine to perform the indicated required function. These functions are used in generating a CCW or updating information needed to construct an IORCB. Overall these functions include allocating space in the IORCB buffer area and moving into this buffer:

- Begin and/or end control characters
- Space for response characters
- Output data
- Data translation of output data
- Space allocation of output data

The last buffer allocation flag is an end flag. If it is not set, the developed CCW is then moved (see note below) into the build area of TOS where the command code and flag fields are obtained from the CCWG area and the count and relative displacement address fields are obtained from registers. The logical function code is moved into the logical function area of TOS to be used later by TAM Posting. The next sequence cycle in this CPG then repeats, by sequentially testing the next CCWG buffer allocation flags.

If the last buffer allocation flag is set, TAM Read/Write prepares to terminate.

Note: A developed CCW is not moved into the build area of TOS and the logical function byte code is not moved into its respective location if the following conditions are present in the CCWG:

- Command code field is zero
- End allocation flag is set
- Inhibit allocation flag is set

TAM Read/Write then prepares to terminate since all the allocation flags in all the CCWGS in the CPG have been tested. To accomplish this termination:

- The message in the buffer area of the IORCB is translated to the terminal character set code, for all write operations.
- The CCW list is moved from the TOS build area into the IORCB CCW list area.
- The IORCB fixed area is completed.

Table 9. TAM Read/Write: Type Option (Hex and Mnemonic) Codes and Description (Part 1 of 2)

Option Code (Hex)	Mnemonic Option Code	Description
02	TID	Read Initial with Dial: This option indicates that an automatic dial connection is to be made with the terminal. The dialing digits are located in the terminal entry list (DFTRMENT). If the terminal type requires polling, the necessary polling sequence characters are generated.
03	TDR	Read initial with dialing/repeat.
04	TIN	Read Initial: It assumes that the line connection has been previously made. If the terminal type requires polling, the necessary polling sequence characters are generated.
05	TNR	Read initial/repeat.
06	TCN	Read Continue: This option is specified when polling is not required. It may be used for terminals previously polled and in a transmit state.
07	TCR	Read continue/repeat.
08	TID	Write Initial with Dial: This option indicates that an automatic dial connection is to be made with the terminal. The dialing digits are located in the terminal entry list (DFTRMENT). If the terminal type requires addressing, the necessary addressing sequence characters are generated.
09	TDR	Write initial with dialing/repeat.
0A	TIN	Write Initial: It assumes that the line connection has been previously made. If the terminal type requires addressing, the necessary addressing sequence characters are generated.
0B	TNR	Write initial/repeat.
0E	TIA	Write with Response: It assumes that the line connection has been previously made. If the terminal type requires polling and addressing, the necessary sequence characters are generated. This option provides the ability to output a message to a terminal and receive the terminal's next input record or line. The maximum output message size is 1 to 32,767 characters. The maximum input message is one logical record or line as specified by terminal type.
0F	TAR	Write with response/repeat. The above types with the repeat option will automatically retransmit or request retransmission of messages in error if the terminal is equipped with error correction facilities. A predetermined number of retries as specified by terminal type will be attempted for each message transmitted in error. The posting of uncorrectable errors will include appropriate error information.
64	AUTOWRAP	On accepting this order, the Transmission Control Unit wraps the output of the addressed line to the input of line 0. The command within the channel operates as a write.
65	DISABLE	On accepting this control order, the Transmission Control Unit resets the enable latch within the line adapter of the addressed communications line. No data transfer occurs.
66	ENBLASYN	On accepting this control order, the Transmission Control Unit sets the enable latch within the line adapter of the addressed communication line. No data transfer occurs.

Table 9. TAM Read/Write: Type Option (Hex and Mnemonic) Codes and Description (Part 2 of 2)

Option Code (Hex)	Mnemonic Option Code	Description
67	ENBLSYN	On accepting this control order, the Transmission Control Unit sets the enable latch within the line adapter of the addressed communication line. No data transfer occurs.
68	PREPARE	This order may be used in a contention type communications system to indicate to the processor when data is arriving. When a valid start bit is detected by a line instructed to prepare, a character is strobed off. If at stop time the line is at mark, the prepare command is terminated with channel end and device end status. The character assembled is not transferred to the multiplexor channel. If the line is at space, a timeout is started. If the line returns to mark before the timeout is complete, the prepare command is terminated with channel end and device end. The prepare command is terminated when the timeout occurs, indicating an open line condition with channel end, device end, and unit check status and intervention required in the sense byte.
69	SADONE	On accepting this control order, the Transmission Control Unit sets the terminal control (TC) field within the addressed LCW to one, so that the terminal control with the internal address equal to one is associated with the addressed communications line. No data transfer occurs.
6A	SADTWO	On accepting this control order, the Transmission Control Unit sets the TC field within the addressed LCW to two, so that the terminal control with the internal address equal to two is associated with the addressed communications line. No data transfer occurs.
6B	SADTHREE	On accepting this control order, the Transmission Control Unit sets the TC field within the addressed LCW to three, so that the terminal control with the internal address equal to three is associated with the addressed communications line. No data transfer occurs.
6C	SADZER	On acceptance, the Transmission Control Unit will set the TC field within the addressed LCW to zero so that the terminal control with the internal address equal to zero is associated with the addressed communication line. No data transfer occurs.
6A	BREAK	On accepting this order, the addressed line transmits a continuous space signal. Bytes transferred from the channel to the addressed unit must be all zeros. To provide control over the length of space signal, a byte count must be sepecified by the program.

Table 10. TAM Read/Write: Unit Type Table Format

2702 TERMINAL LIBRARY TABLE (TLT) ADDRESS
2701 TLT ADDRESS
MULTIPLEXER TLT ADDRESS
SELECTOR TLT ADDRESS
2701 OR SELECTOR TLT ADDRESS
Note: Unit Type Table (UTT) is specified in READ/WRITE.

Table 11. TAM Read/Write: Terminal Library Table Format (for 2702 TLT)

1050 TERMINAL CHANNEL PROGRAM (TCP) ADDRESS
2741 TCP ADDRESS 35 TCP ADDRESS

Table 12. TAM Read/Write: Terminal Control Program Format

TERMINAL CONTROL INFORMATION TABLE
CHANNEL PROGRAM INDEX TABLE (CPIT)
CPG DISPLACEMENT FOR TID* (R)
CPG DISPLACEMENT FOR TIN* (R)
CPG DISPLACEMENT FOR TCN* (R)
CPG DISPLACEMENT FOR TID* (W)
CPG DISPLACEMENT FOR TIN* (W)
CPG DISPLACEMENT FOR TCN* (W)
CPG DISPLACEMENT FOR TIA* (W)
CHANNEL PROGRAM GENERATOR for TID* (R)
CPG FOR TIN* (R)
CPG FOR TCN* (R)
CPG FOR TID* (W)
CPG FOR TIN* (W)
CPG FOR TCN* (W)
CPG FOR TIA* (W)
*All CPG displacements need not be required for the terminal type. These displacement fields are zeroed and the corresponding CPG fields do not exist.

Table 13. TAM Read/Write: Selected Terminal Control Information Table Entries

Maximum Option Amount; Begin Control Character Count; Begin Control Character.
Maximum Character Set Amount; Polling Character Count; Polling Characters.
Prefix Count; End Control Character Count; End Control Characters.
Read End Of Line Sequence Count; Read End Of Line Sequence Character.
Maximum Number of CCW; Addressing Character Count; Addressing Characters.
TAM READ Data Set Length; Control Character Table Address.
Write Error; Read Error; Read Error Positive Response.
Control Count; Positive Response; Characters.
Maximum Count Of Line Control Characters; Read Error Negative Response; Read Error Negative Response Characters.
TAM WRITE Data Set Length.
Polling Response; Addressing Response Count; Addressing Response Characters.
Repeat Option; End Of Line Sequence Count; End of Line Sequence Characters (20 Characters Maximum).
Maximum Buffer & CCW Size; End of Card Sequence Count; End of Card Sequence Characters.
Terminal Character Set Address; Translate and Test Function Table Address.
Continue Count; Text Control Character Count; Text Control Characters.
Failure Count; End of Message Sequence Count; End of Message Sequence Characters (20 Characters Maximum).

- An IOCAL is issued, which links to the supervisor to initiate the terminal computer communication, under control of the IORCB CCW list and through the IORCB buffer area.

When the supervisor returns control, TAM Read/Write restores the registers and issues a return to the user, if the posting flag is not on.

If the IORCB buffer data area was not large enough to complete the TAM Read/Write operation, it is necessary for continuous IORCBs to be developed for one DECB. Although the initial TAM READ/WRITE was from the user, subsequent IORCB entries would be from TAM Posting until the operation is complete. These TAM Posting calls to this program cause a return to TAM Posting under direction of the posting entry flag in TOS.

IOREQ Routine (CZCSB)

During the execution of a user's program, where the programmer had originally issued an IOREQ macro instruction to generate an I/O operation on a device, a call is

Table 14. TAM Read/Write: Channel Command Word Generator Section

Name	Comments
Command Code Field	Command code for a particular operation. This byte is moved into the command code field of a CCW list located in the build area of TOS.
Unused Field	This field is reserved for future expansion.
Displacement Field	Displacement of ending point to CCWG when completing a CCW or a regeneration of a particular command generator when multiple IORCBs are required.
Flag Field Logical Function Code Field	Flags required for a particular command generator. Required for TAM Posting routine. The byte is moved into the logical function area of TOS.
Buffer Allocation Flag Field	Required in building the CCW. Allows additional routines to be called to fill in the buffer area of the IORCB when a CCWG is processed.

Table 15. TAM Read/Write: Channel Command Word Generator Format

Command Code	See Table 9.
Displacement	Completed by IOS.
Flags	Data chaining, command chaining, suppress length indication, skip, program control interruption.
Logical Function Code	Dial end control, data out, data in, read error response, write error response, addressing, polling, address response, polling response, control, end control, error TIC, negative response, write error message.
Buffer Allocation Flag	See Table 16.

generated that links to IOREQ. A system routine may also directly call IOREQ at a second entry point.

IOREQ uses control blocks and buffer areas that are allocated during IOR OPEN to build channel command words in the IORCB. (See Chart BE.)

Attributes: Reentrant, resident in virtual storage, closed, nonrecursive, read-only, privileged.

Entries:
CZCSB1 -- Type-2 linkage. Entered from user IOREQ macro instruction.

CZCSB2 -- Type-1 linkage. Entered call by system.

Input: Register 1 contains the address of a two-word parameter list:

Word 1 -- Address of the DECB.

Word 2 -- Address of main storage furnished by the privileged user for IOREQ portion of DCB and for IORCB.

The second parameter word is used only for entry at CZCSB2.

Data Reference: CHADEC, CHADCB, CHADEB, CHAISA, CHAIOR.

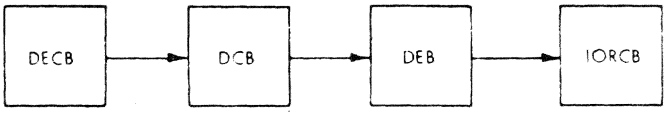
Modules Called:
CKCLS (CEAQ4) -- Check protection class.

IOCAL (CEAA0) -- I/O call.

Exits:
Normal -- Return to calling routine.

Error -- ABEND macro instruction.

Operation: A user originally issuing an IOREQ macro instruction enters IOREQ to edit the VCCW list. IOREQ initially saves the general registers and then loads them with pointers to referenced control blocks. This starts with the DECB.



The DECB pointer to the DCB is included as one of the IOREQ macro instruction operands, and the other pointers are described in IOR OPEN. IOREQ then performs the following functions:

- Validates user parameters.
- Checks that the DCB identification is valid DCBID=*%*%.

Table 16. TAM Read/Write: Buffer Allocation Flag Bits of CCWG

Bits	Name	Comments
48	Unused	Reserved for future expansion.
49	Begin Control	0 - No control characters required. 1 - Begin control characters required before polling or addressing is initiated. Characters are obtained from the TCI and moved to buffer area in CHAIOR.
50	End Control	0 - No control characters required. 1 - End control characters required after polling or addressing has been initiated. Characters are obtained from the TCI and moved to buffer area in CHAIOR.
51	Text Control	0 - No control characters required. 1 - Text control characters required after end control characters. Characters are obtained from the TCI and moved to buffer area in the CHAIOR.
52	Data Out	0 - No data transmitted to terminal. 1 - Data to be moved from user area (address in CHADECO) to buffer area in CHAIOR (data is tested for function control characters). Function Control Characters and related codes which are not recognized by different terminal types are deleted from message when moved to the buffer area in CHAIOR. Translating data to terminal character set code is done at completion of CHAIOR.
53	Data In	0 - No data transmitted to CPU. 1 - Data to be transmitted from terminal to CPU. A buffer area is reserved in the CHAIOR to accept the message. The message is not tested for function control characters. Translating data to EBCDIC is accomplished in TAM POSTING routine.
54	Read Error Positive Response	0 - No Read Error Positive Response characters required. 1 - Read Error Positive Response characters required after transmission of data to CPU. Characters are obtained from the TCT and moved to buffer area of the CHAIOR.
55	Read Error Negative Response	0 - No Read Error Negative Response characters required. 1 - Read Error Negative Response characters required after transmission of data to CPU. Characters are obtained from the TCI and moved to buffer area of the CHAIOR.
56	Write Error Response	0 - No Write Error control characters required. 1 - Write Error control characters are transmitted by the terminal control unit. The count is obtained from the TCI and space allocated in the buffer area of the CHAIOR.
57	Addressing	0 - No addressing required for a terminal. 1 - Addressing required, obtain addressing characters from the terminal entry list (DFTRMENT-address located in DECB). If not present, addressing characters are obtained from the TCI. Characters are placed into the buffer area of the CHAIOR.
58	Polling	0 - No polling required for a terminal. 1 - Polling required, obtain addressing characters from the terminal entry list (DFTRMENT-address location in DECB). If not present, polling characters are obtained from the TCIB. Characters are placed in the buffer area of the CHAIOR.
59	Addressing Response	0 - No addressing response required. 1 - Addressing required, obtain characters from the TCIB. Characters are moved into the buffer area of the CHAIOR.
60	Polling Response	0 - No polling response required. 1 - Polling required, obtain characters from the TCIB. Characters are moved into the buffer area of the CHAIOR.
61	Inhibit	0 - Current channel command word will be developed and placed into the build area of the CHATCS. 1 - Inhibit current channel command word from being developed.
62	Continue	0 - Follow normal sequence of CCWG operation. 1 - Normal sequence of CCWG is changed. It permits return to a previously executed CCWG.
63	End	0 - Continue normal sequence of CCWG. 1 - Terminates the CCWG list.

- Checks that the DEB identification is valid DEBID=*(.
- Stores address of DECB in DCBTMP (work area).
- Clears the fixed portion of the IORCB.
- Clears the outstanding IOREQ requests (DECB Queue) if the DEB flag (DEBNF7) is set. This indicates that the previous CHECK macro instruction caused an exit to SYNAD. Therefore while executing the previous SYNAD routine the user must reissue desired purged DECBs.

Note: If either the DCB or DEB identification is not valid, an exit to ABEND occurs.

Pre-edit checking now occurs and, if satisfactory, IOREQ begins the edit phase. This pre-edit checking includes the following:

1. Check if the IORCB can be executed. If a previous operation set the DEB flag (DEBNF1) to indicate an intercepted error, the IORCB is marked intercepted and a return occurs. The IORCB cannot be executed at this time, but the DECB is entered in the queue and the IOREQ should be reissued by the user.
2. Check if the DECB is valid. A return occurs if the DECB has any of the following invalid conditions:
   - The DECB is in the wait state (DECB not ready).
   - The DECB is in use (DECB active).
   - The queue of pointers in the DEB is full (DEBNPC is exceeded).

IOREQ then begins the edit phase. This edit phase consists of a preliminary pass through the VCCW list pointed to by the IOREQ macro instruction to determine the validity of the VCCWs. In addition, the amount of space required for the buffer or page list entries and the VCCW list is determined.

If buffering is requested by the IOREQ macro instruction, the following pointers and counters are also initialized:

- A pointer to the lowest address of a read request VCCW, used to determine the low-order buffer address needed for the read request.
- A pointer to the highest address of a read request VCCW, used to determine the high-order buffer address needed for the read request.
- A counter containing the amount of buffer space needed in the IORCB for read requests. For these read requests, the amount of buffer space is the contiguous difference between the high and low-address buffer pointers.

Tests are also made in this edit phase to assure that chaining rules are not violated. Chaining rules are listed in IOREQ: VCCW section of Assembler User Macro Instructions, GC28-2004. If any rules are violated, a return to the user occurs. The user's program has been checked for validity within IOREQ but a system program that calls IOREQ directly, enters at this second point and therefore is assumed to have the VCCW list built correctly without the need of editing. However, on IOREQ recognizing a system program entrance, it branches to the previous phase only to set up buffer addresses and values or to build page list entries. During this build phase a different sequence is followed depending on whether or not buffering is specified.

If buffering is requested:

- The start address of the IORCB buffer is set.
- The buffer total length is set.
- The start of the CCW list is set.

The CCW entries are generated in the IORCB using the VCCW list as input, and space is allocated in the buffer area. In addition, in write and control requests, data specified in the fields of the current VCCW are moved into the buffer. If the CCW is a transfer-in-channel (TIC), the displacement address of the CCW is set to point to the desired CCW (displacement is from CCW origin).

If buffering is not requested by the user, the appropriate addresses in the IORCB are set from values calculated in the edit phase (the previous pass through the VCCW list). The number of page list entries is calculated for the privileged request by a special branch to the initial edit section. For read, write, and control requests, the linkage is made from the CCW page list pointer to the appropriate page list entry, by searching through the page list for an address which has the same segment and page in the CCW.

For both buffered and nonbuffered IORCBS, the following also occurs:

- The length of the IORCB is obtained by adding to the fixed area, the buffer area (either buffer or page list entries), and the CCW list length.

- The length of the IORCB, if greater than the maximum allowed, causes IOREQ to return to the user with an error code in register 15.

- This pass through the VCCW list moves the op-code count, and flag fields to the appropriate CCW list entry in the IORCB.

- The protection key and other flags are set in the IORCB.

The system programmer that entered at the second entry point now returns with a return code of zero in register 15. Hence to execute his IORCB, the system programmer must update the required fields and issue his own IOCAL. The sequence IOREQ follows for the IOREQ macro instruction user is:

- Test the IOC flag in the last VCCW. If set, this IORCB must be chained to the next IORCB.

- Move the DECB address into the queue by updating pointers to this DECB as well as updating the number of IORCBs and DECBs that are outstanding.

- Determine if the IORCB is not to be executed so that the DECB can be placed in queue and marked intercepted.

- Issue the IOCAL macro instruction.

- Return to the user.

POSTING AND CHECK PROCESSING

The following routines describe posting and error processing for SAM, SAM direct access, MSAM, TAM, and IOREQ, and the operation of the Check routine.

SAM Posting and Error Retry Routine (CZCRP)

This routine processes a synchronous I/O interruption caused by the termination of a SAM I/O operation. It posts the completion code in the event control block (ECB) of a data event control block (DECB). In addition, depending upon the device, this routine may perform other post-I/O activities, such as adjusting magnetic tape block counts. The retry/recovery operations are also incorporated into this routine; they are employed if the channel status word (CSW) returned at the most recent interruption reveals an abnormal end condition. (See Chart CA.)

Attributes: Reentrant, nonrecursive, resident in virtual storage, privileged.

Entry Point: CZCRP1 -- Main entry point via type-1 linkage.

Input: The channel status word (CSW) and sense information pertinent to the I/O operation are contained in the ISA.

Data References: CHADEC, CHADEB, CHADCB, CHAIOR, CHAISA, CHASDA, CHASDT.

Modules Called:
VMER (CZRX2) -- Virtual memory error recording.

VMSDR (CZCRYY) -- Virtual memory statistical data recording.

ABEND (CZACP1) -- Abnormal task termination.

QLE (CZCJTQ) -- Build queue in task monitor.

WTO and WTOA (CZABQ1) -- Write message to operator.

ASCII Translation (CZCWA1) -- Translate ASCII data to EBCDIC.

DA Error Retry (CZCRH1) -- Direct access error retry routine.

Reset (CEEAH) -- Permit task to access I/O device.

SYSER (CEAIS) -- System error processor.

Exits:
Normal -- Return to task monitor.

Error -- An ABEND is issued if the I/O device is not tape or disk. A SYSER is issued if:

- A program check has occurred.
- A protection check has occurred.
- A specification error has occurred.
- The CSW has a status of X'00'.
- There was no entry for the SDA.
- A command reject occurred because of an invalid op code.

Operation: SAM Posting examines the completed IORCB and either posts the DECB or passes control to a device-dependent error routine.

Completion of the requested I/O is signalled by the device via an I/O interruption that is associated with the IORCB by the Supervisor.

The completion information, along with the IORCB, is passed back to the requesting task as a synchronous I/O interruption enqueued on the TSI. During the interval between request and response, the task could have generated other I/O requests, and possibly had its time slice ended.

SAM Posting requires that all interruptions be masked. It will process a synchronous I/O interruption to completion, regardless of the interrupted operation. SAM Posting will process only one synchronous I/O interruption at a time for a given task.

The I/O Supervisor is expected to return unexecuted IORCBs when there is an I/O interruption caused by hardware failure, a unit check, or a unit exception condition.

SAM Posting expects the following fields of the CHADEC Table (DECB) to be zeroed: ECB, BSJ, SB1, SB2, FLG, CSE. Under error conditions, it examines BSF to determine how many sense bytes from the ISA are to be saved for user reference. These are placed

in the last field of the DECB (normally BSF will contain X'02', indicating all eight sense bytes are to be saved).

For the Control routine (CNTRL), IORCB, Rewind, and Rewind and Unload, the DECB pointer in the IORCB will equal zero. The Check routine will clear the Error flag in the DEB to zero upon return from the user's SYNAD routine.

In handling the I/O synchronous interruption, the task monitor performs short save to provide working registers, and executes a type-1 call for the appropriate posting and error recovery routine. SAM Posting is the proper routine for SAM.

SAM Posting first determines if the IORCB has been executed. An IORCB would not be executed if the preceding I/O operation for that device had resulted in an abnormal completion. (Such a check must be done to handle error conditions properly when multiple IORCBs can be outstanding for one device by one task.) If the IORCB was not executed, the IORCB is reissued (via IOCAL), or the DECB is posted to indicate "Intercepted" status. The choice depends upon whether the preceding condition was a permanent error.

NORMAL COMPLETION: If the IORCB was executed, a test for normal completion is made. If normal, the event control block of the DECB is posted to indicate normal completion. SAM Posting also performs other optional services which can only be done after completion of the I/O operation:

- Data Movement - Movement of data from an IORCB buffer to the user area for an input operation can be requested by the user.
- Magnetic Tape Block Counts and Unit Exception Flag - SAM Posting increments (for forward tape movement) or decrements (for backward movement) the proper counts. The routine also sets appropriate flags for unit exception conditions.
- Direct Access Read Variable Length Record and Pending C.W.E. Flags - The routine will perform the appropriate address movement and flag setting.

OTHER THAN NORMAL I/O COMPLETION: IOS will lock the device to further I/O by this task, until a Reset SVC is issued by SAM Posting. This Reset SVC, when issued, will set off the suppression flag so that the task may resume I/O activity on the device in question.

If the I/O was not normally completed, further tests are made and control is passed, if necessary, to device dependent error recovery routines which, if possible, will issue a retry I/O request. Each of these device dependent error recovery routines subsequently returns to the posting routine.

Unrecoverable (Hardware) Error Completion - General (non-device-dependent) Processing: If the CSW information associated with the executed IORCB does not indicate completion, the routine searches for an immediate hardware indication of unrecoverable error.

If the error or condition is not yet considered to be unrecoverable, the routine will issue a Reset SVC to IOS. The purpose of the Reset SVC is to "unlock" the device queue previously "locked" by the I/O Supervisor.

If the error is unrecoverable, the appropriate flags, addresses and counts are set.

Recoverable Error or Exceptional Condition (non-Error) Completion - General Processing: This condition is indicated by the Unit Check or Unit Exception flags.

Recoverable Error or Exceptional Condition Completion - Device-Dependent Processing: If possible, depending on the device, N retries over the original path and also N retries over 3 alternate paths will be made, until a successful retry or until the set maximum of retries is reached. In the latter case the failure is termed a "hard" failure. The VMER routine is called to record data associated with this "hard" failure.

Appropriate flags are set, messages are put out, counters are incremented, information is saved, and routine linkage is effected.

Device-Dependent Error Procedures: The following are general notes on the device dependent error procedures:

Error indication: When certain malfunctions occur, the CSW will contain more than one error indicator. Generally, only one of these properly describes the malfunction while the other(s) indicate secondary effects. Similarly, some device/control Unit errors can cause more than one sense bit to be present.

Original and alternate path retries: For some error conditions, there are no original path retries. However, there are always alternate path retries. An original path retry utilizes the same channel and device as that used in the original erroneous I/O

operation. An alternate path retry uses the same device, but goes through a different channel.

Number of retries: The number of retries is dynamic; the installation, however, not the user, determines the threshold number. In the descriptions of the error retry procedures, in most cases N = a + b where a is the threshold number, and b is a constant for that particular approach to recovery.

User's SYNAD routine: If appropriate, SAM Posting will set the SYNAD flag in the user's DECB. The user's SYNAD routine is his own "error" routine. It does not, however, attempt error recovery on I/O devices, but rather determines if the user wants to terminate or continue processing.

Return codes from error recovery routines:

00 - Retry in progress.

04 - Permanent error encountered, no retry.

08 - Normal completion.

0C - Complete with error without retry.

10 - Complete with error after retry.

Rebuild of the IORCB:

The IORCB will be in page 0, segment 0 of virtual storage (that is, within the interruption storage area). The issuance of an error retry IORCB will consist, in the main, of initializing certain fields automatically, conversion of the CCW list from real core to virtual storage addresses, appending of the CCW list if required, and initialization or modification of other fields within the IORCB. The purpose of these actions is to provide IOS with an input IORCB, that is, an IORCB with no main storage references, and no past action flags set (such as start I/O failure).

The need for appending a channel program with additional CCWs is determined in accordance with error retry requirements. The appendage will be made to the end of the channel program. If additional CCWs are added to the IORCB, the following fields will require incrementing or other modification:

1. IORLN -- IORCB length; calculate the new value in accordance with the number of additional CCWs in the appended list.

2. IORCL -- CCW list length; increment by the number of additional CCWs appended.

3. IORST -- Relative origin of "Start CCW". Modify this field in accordance with the error retry procedure.

4. IORSC -- IORCB Software Command Chain flag set if addition to the CCW list requires it.

MAGNETIC TAPE - 2400 TAPE SERIES ERROR RETRY PROCEDURE: The device-dependent error procedure for the 2400 tape series is described as follows:

Unit Check (CSW bit 38): The routine checks the sense byte information to determine the cause of the unit check condition. Sense byte information will be found in segment 0, page 0 of virtual storage. Next, the routine checks the corresponding bit position in the IMSK field of the DCB.

Load Point Sense Bit Only: The routine will set Complete With Error in the ECB, set on the Error flag, and move the CSW and the first two sense bytes from segment 0, page 0, to the DECB. The tape block count will be decremented by 1 and DEBMSK in the DEB will be incremented by 1.

Load Point and/or Other Sense Bits Set:

1. Bus Out Check (Byte 0, bit 2) -- The retry will consist of repositioning the tape, if required, and a repeat of the failing CCW. If tape motion takes place and the failing operation is a forward or backward space of record or file the error will be deemed unrecoverable. The routine will set the Permanent Error indication in DCBIFL, set the Permanent Error flag in the DEB, and provide a return code of '04'.

2. Equipment Check (Byte 0, bit 3) -- Bit 7 of sense byte 3, or one or more of the bits of byte 4, will also be set to give more detail about the hardware failure. There is no original path error retry. Since data transfer and tape motion are indeterminate for all equipment checks, there will be no alternate path error retry. The routine will set the Permanent Error indication in DCBIFL, set the Permanent Error flag in the DEB, provide a message to the operator, and provide a return code of '04' in general register 15.

3. Intervention Required (Byte 0, bit 1) -- If the addressed tape unit is nonexistent, (indicated by sense byte 1,

bit 2 equal to zero), the routine will try an alternate path.

The routine will also perform ABEND processing for a control routine IORCB when the operation is other than rewind and unload (RUN). If the operation is RUN then a return code of zero will cause a return to the task monitor. For a non-control routine IORCB, if the selected tape unit is in the end-of-tape area, the routine will set Request for Synad in the DECB and the Error flag in the DEB. It will also set Complete With Error in the ECB, move the CSW to DECCSW, move the first two sense bytes to DECSB0 and DECSB1, and all eight sense bytes to DECASB. The routine will then set a return code and a message will be sent to the operator to demount the tape. If the selected tape unit is not in the end-of-tape area, execution of the CCW list will be resumed from the point of interruption. A message will be sent to the operator to ready the tape unit, and the routine will set a return code of zero after a return from the IOCAL routine.

4. Overrun (byte 0, bit 5) -- Data transfer will be stopped. Retries over the original path will consist of repositioning the tape and re-initiating the failing command. This error should not be associated with a control operation and, if such should be the case, the routine will request an abnormal termination of the task. If the error retries fail, the routine will send a message to the operator.

5. Data Check (byte 0, bit 4) -- Bits 0 through 3 of byte 3 will be set to give more specific detail regarding the data check.

   a. Control Operation -- If the failing operation is write tape mark (WTM), then original path error retry will be attempted N times. The error retry procedure will consist of backspacing the tape one block, followed by an erase gap command, and then a repeat of the failing write tape mark operation. If the error retries fail, the routine will send a message to the operator.

      If the failing operation is other than WTM, the original path error retry will be attempted N times by retrying the command which failed.

   b. Write Operation -- Same as first paragraph of a.

   c. Read Operation -- If the operation was a read backwards and if the tape is at load point, spurious noise was detected and the data check should be ignored. In this instance the routine will set on the DEB error flag, mark the ECB Complete With Error, and move the CSW into DECCSW and sense bytes into DECSB0 and DECSB1. If this was an error retry IORCB, the routine will set off the Error Recovery In Progress indication and set a return code in general register 15. If this was not an error retry IORCB, the routine will set a return code in general register 15. If the tape is not at load point, or if it is at load point but the operation was not a read backward, then a test is made to see if the block meets minimum block length requirements.

      If this is a noise block, and the interruption occurred on the last CCW, the routine will set a return code for Normal Completion termination processing in general register 15. If more CCWs remain, the routine will rebuild the IORCB, and will resume the channel program from the point of interruption. Whether the record was a noise record or not, the read will be retried N times. The error correction programming sequence consists of setting the correct mode, repositioning the tape, sending track-in-error information to the control unit, and then issuing a read or read backward command. An error that persists should cause the tape to be backspaced five blocks (if five are available), thus placing the tape past the tape cleaner. An attempt is again made to read the tape, using the procedure just described. This loop should be repeated until the error is corrected, up to a maximum of N reads. Should the error still persist, the associated block is defined as a permanent read error. The routine sends a message to the operator indicating an unrecoverable error.

6. Data Converter Check (byte 0, bit 7) -- If the chaining check bit is on, the action in the subsection referring to CSW bit 47 below is performed. If the chaining check bit is not on, the routine will abnormally terminate.

7. Command Reject (byte 0, bit 0) -- If the tape is file protected and the channel command word was a write, write tape mark, or erase gap, the

DECID is set to X'40', an unrecoverable error indication is set, SYNAD is requested, the error is recorded through the VMER macro, and return is made. If the tape is not file protected, an exit is made via SYSER.

Chaining Check (CSW bit 47): If the failing operation is a read command, the original path error retry is attempted N times. The error retry consists of repositioning the tape and reinitiating the command that failed. If the error persists, the routine provides a message to the operator indicating "unrecoverable error". The routine will also set on the "Permanent Error" flag, set on the "Permanent Error" indication, and move the channel status byte from the CSW to the IORCB. The routine will send a return code of '04' in general register 15.

If the chaining check occurs on other than a read operation after N error retries, the routine:

(1) Sets the Error and Permanent Error flags on.

(2) Sets DCBIFL to Permanent Error condition.

(3) Sets DECECB to Complete With Error.

(4) Sets Synad Requested in the DECB.

(5) Moves CSW from ISA to IORCB.

(6) Interfaces with the VMER routine.

Unit Exception (CSW bit 39): This bit is set to indicate a read of a tape mark, or a write in the end-of-tape area. If the CCW involved in this interruption is an erase gap (ERG) and there are remaining CCWs, the routine sets the IORUE flag on, rebuilds the IORCB, and resumes the CCW list with the remaining CCWs. If this interruption occurred on an error retry within an appended CCW, the routine rebuilds the IORCB and restarts with the original failing CCW.

If this was not an error retry, or if the interruption did not occur on an appended CCW, or if the CCW involved was an ERG with no remaining CCWs, the routine will process as follows:

(1) Set on the Error flag.

(2) Set DECECB to Complete With Error.

(3) Move the CSW to DECCSW.

(4) Move the first two sense bytes to DECSB0 and DECSB1 and all eight sense data bytes to DECASB.

Incorrect Length (CSW bit 41): If the failing CCW is a read operation the routine will determine if the record length conforms to one of the following standards:

For fixed block the record length equals a multiple of the block length.

For variable length the CCW count equals the residual count plus LL. For an ASCII variable-length record the CCW count equals LL plus the buffer offset minus 4.

An undefined length is automatically acceptable, and a "fixed standard" length is an automatic error.

If acceptable, "normal completion" is set and a return occurs. If in error, "SYNAD request" and "complete with error" are set in the DECB, "unrecoverable error" in the DEB, the sense bytes and CSW bytes are saved in the DCB, and a return occurs.

DA Error Retry Routine (CZCRH)

DA Error Retry processes synchronous I/O interruptions originating from a SAM or an Obtain/Retain operation on a DA device. DA Error Retry modifies the channel program, rebuilds the IORCB, and reissues the I/O request (see chart CB).

Attributes: Privileged, reentrant, closed, resides in virtual storage.

Entry Point: CZCRH1 -- Entered via type-1 linkage.

Input: When this routine is entered, the channel status word and sense information pertinent to the I/O operation are contained in the ISA.

Data References: CHASDT, CHAISA, CHAIOR, CHADEB, CHADCB, CHADEC.

Modules Called:

VMSDR (CZCRY) -- Virtual memory statistical data recording.

VMER (CZCRX) -- Virtual memory error recording.

VMA (CZCGA) -- Get work area.

SIR (CZCJS) -- Select interruption request (handle asynchronous I/O).

QLE (CZCJT) -- Build queue in task monitor.

WTO (CZABQ) -- Send message to operator.

SPATH (CEAAB) -- Mark the device, control unit, or channel, down or OK.

STRCT (CEAH3) -- Get task ID.

SETAE (CEAAK) -- Set up interruption queue.

Exits:
Normal -- Return to SAM Posting with the return code in register 15.

Error --

- Abnormal termination via the ABEND macro instruction.
- SYSER.

Operation: For general notes on device-dependent error procedures, see the SAM Posting and Error Retry routine, "Operation" section, under "Device-Dependent Error Procedures."

For each error retry attempt the DA Error Retry routine increments the appropriate error retry counter by 1, sets the Error Retry flag on in the IORCB, sets the Error flag on in the DEB and sets the Error Recovery in Progress indication in the DCB. The IORCB is rebuilt appropriately and executes the IOCAL SVC preparatory to re-execution of the channel program. A return code of zero is set in general register 15 prior to returning to the calling routine.

For each unrecoverable (permanent) error the routine sends a message to the operator indicating unrecoverable error, sets on the Permanent Error flag, sets the Permanent Error indication in DCBIFL and moves the channel status byte from the CSW in the ISA to IORCSB. When the error source is indicated by the channel status byte, this routine interfaces with the VMER routine, sets a return code of '04' in general register 15, and returns to the calling routine.

When the DCBIMK corresponding bit is zero (that is, the user does not wish to have a retry for this error), the routine will move the CSW from the ISA to DECCSW, move sense bytes 0 and 1 from ISA to DECSB0 and DECSB1 respectively, and set return code of '0C' in general register 15.

The following shows termination processing to be done by the SAM calling routine after the direct access retry routine sets a return code in general register 15 other than zero. For a return code of zero (which occurs when the error recovery procedure issued an Error Retry IORCB) the calling routine will exit to the task monitor without any further processing.

Action	Return Code in G.R. 15				
	04	08	0C	10	14
Set DECECB=X'41'=Complete With Error	✓		✓	✓	
Set Synad Requested flag in the DECB	✓		✓	✓	
Move CSW to DECCSW	✓	✓			
Move sense bytes 0 and 1 to DECSB0 and DECSB1	✓	✓			
Set Error flag, DEBNF1, to 1	✓		✓	✓	
Decrement DEBIOC by 1 (Allowed IORCB o/s Count)	✓	✓	✓	✓	
Move data from IORCB to user virtual storage, if required	✓	✓	✓	✓	
Device-dependent termination processing	✓	✓	✓	✓	
Normal Completion ='7F' set in DECECB		✓			
Link to VMSDR -- after successful error retry		✓		✓	
Link to VMER			✓	✓	✓
Clear DEBNF1 to zero after successful error retry		✓			
Set DCBIFL Error Recovery in Progress indication=0 after successful error retry		✓			

CONTINGENT ADDITIONAL PROCESSING: In addition to that processing which occurs with each error retry attempt, the following contingencies may be encountered and the ensuing processing involved.

Channel Data Check (2311, 2314, 2302): Original path error retry is attempted. Each time an alternate path is requested, the channel status byte of the CSW will be moved from (0, 0) to IORCSB in the IORCB to be used by the VMER routine.

Unit Exception: The Error flag is set on and a test is made for a corresponding DECB.

1. If a corresponding DECB exists, DECECB is set to indicate Complete With

Error, the CSW is moved from the ISA to DECCSW, and two bytes of sense information are moved from the ISA to DECSB0 and DECSB1.

2. If a DECB does not exist and this is not a retry IORCB, a return code of '0C' is set in general register 15 before exit. Otherwise, when a DECB does not exist and this is a retry IORCB, a return code for VMSDR interface is set prior to exit.

Chaining Check: There will be N original path error retry attempts consisting of repeating the original CCW list.

Incorrect Length: The action taken in this case is contingent upon the format of the record being processed as indicated in DCBREC. The possible logical paths and actions performed are:

1. Unknown - normal completion is indicated by a return code of '08'.

2. Variable - if the residual count of CSW is equal to zero, Complete With Error is set in the DECB, the SYNAD Requested flag is set, the Error flag is set in the DEB, the CSW is moved from (0, 0) to the DECB, and the return code is set to '0C'.

If the residual count of the CSW is not equal to zero, a comparison is made between zero, and the difference obtained by subtracting the sum of the CSW residual count and the 'LL' count from the CCW count.

If the result of subtraction is 0, the return code is set to '08'; and Normal Completion processing will take place. If the result of subtraction is not 0, "Complete With Error" is set in the DECB, the SYNAD Requested flag is set, the error is indicated in the DEB, the CSW is moved from (0, 0) to the DEVB, and a return code of '0C' is set.

Unit Check (2311, 2314, 2302): The number of retries over the original path is dependent upon the type of error which caused the unit check condition. Examination of the sense byte data will indicate the error cause. Prior to any diagnosis or error retry, the appropriate bit in the DCBIMK field of the DCB is tested against the corresponding bit set in the sense byte data. If the bit is off, further processing will be that as contained above for DCBIMK corresponding bit equal to zero. If the bit is set on, diagnosis and retry continues as follows:

Equipment Check (byte 0, bit 3) -- There is no original path error retry. The alternate path error retry will consist of repeating the original CCW list.

No Record Found (byte 1, bit 4) -- When the Missing Address marker (byte 1, bit 6) is also set, there are N original path error retry attempts. For a 2311 the error retry procedure consists of a Restore CCW followed by a TIC to seek the original address.

If the Missing Address Marker bit is not set, there are N original path error retry attempts. Initially this error retry procedure consists of verification of the home address. This is done by comparing the CCHH of the home address stored in the IORCB against the CCHH of the search argument. If the comparison indicates equality, an ABEND situation is encountered. The routine performs termination processing by setting on the Error and Permanent Error flags, setting ABEND Requested in the DECB, and setting the return code to '0C' in general register 15.

If the CCHH comparison indicates an inequality, the error retry procedure will be as indicated when the Missing Address marker is set.

Seek Check (byte 0, bit 7) -- If Command Reject (byte 0, bit 0) is also set, an ABEND situation is encountered, and a return code of X'12' is set before return. If Command Reject is not set, there are N original path error retry attempts.

Intervention Required (byte 0, bit 1) -- A message will be sent to the operator to ready the device and execution of the CCW list will be resumed from the point of interruption.

Bus Out Check (byte 0, bit 2) -- The original path error retry attempt will consist of repeating the original CCW list.

Data Check (byte 0, bit 4) -- There will be N original path error retry attempts consisting of repeating the original CCW list. After unsuccessful retries, if there is a data check in the count area, the routine abnormally terminates. If there is a data check in the count area, and the Read Variable Length Records flag is set, and the failing CCW is the last in the CCW list, then termination processing will be in accordance with the return code in general register 15 set to '04'. If the foregoing 'AND' situation does not occur the Error and Permanent Error flags, DEBNF1 and DEBNF2, are set, and an ABEND is effected.

When error retry has been exhausted for data check and there is no check in the count area, then Overflow Incomplete (byte 1, bit 7) is checked. If there is an

"Overflow Incomplete" condition, processing will be as follows:

a. Add 1 to 'HH' of seek to seek to next consecutive track.

b. Set R=1 in search argument.

c. Set 'CCHH' of search from 'CCHH' of seek argument.

d. Append CCW list 'B' to original CCW list.

e. Set start CCW to 1st seek of List 'B'.

f. Bookkeep (IORCB) IORCL and IORLN fields.

g. Adjust count and data address of failing CCW.

h. Execute IOCAL SVC after setting flags.

When error retry has been exhausted for data check and there is no data check in the count area, and Overflow Incomplete is not set then:

1. If the Read Variable Length Records flag is set on:

   a. Subtract 1 from 'R' field of search argument.

   b. Append appropriate CCWs to the original CCW list in order to ensure that the correct data area is being searched.

   c. Set Start CCW to first seek of the appended channel program.

   d. Bookkeep IORCL and IORLN fields.

   e. Execute IOCAL SVC after setting flags.

2. If the Read Variable Length Records flag is not on:

   a. For a read set a return code of '04' in general register 15.

   b. Set DEBNF2 and a return code of '04' in general register 15 for a Write operation.

Overrun (byte 0, bit 5) -- There will be N original path error retry attempts consisting of repeating the original CCW list. When the retries are unsuccessful, an ABEND situation is encountered.

Missing Address Markers (byte 1, bit 6) -- There will be N original path error retry attempts consisting of repeating the original CCW list.

Command Reject (byte 0, bit 0) -- This is an ABEND situation.

Track Condition Check (byte 0, bit 6) -- An additional check is made on the Read RO Failed and Alternate Track Indicators. If the read RO failed, this is an ABEND situation.

If the Alternate Track flag is set, the 'CCHH' of the seek argument is set equal to the 'CCHH' of the defective track plus one, and one of the following will occur, depending upon conditions:

1. If the Overflow Incomplete (byte 1, bit 7) bit is not set append CCW list "A" to original CCW list, set start CCW to first seek or "A" CCW list, bookkeep IORCL and IORLN, and Execute IOCAL SVC after setting flags.

2. If the Overflow Incomplete bit is set on, set R=1 in the search argument, set search argument 'CCHH' from seek argument 'CCHH', append CCW list "B" to original CCW list, set start CCW to first seek of list "B", set the IORCL and IORLN fields of the IORCB, adjust the count and data address of the failing CCW, and execute the IOCAL SVC after setting flags.

If the Track Condition Check is set and the Alternate Track flag is not set, the DEBATK and DEBETK 'MBB' are set from the seek argument 'MBB', the DEBATK 'CCHHR' is set from the 'CCHHR' of the alternate track, the DEBETK 'CCHHR' is set from the 'CCHHR' of the defective track, and the seek argument 'CCHH' is set from the 'CCHH' of the alternate track. If Overflow Incomplete is not set, process as in 1 under Track Condition Check. If Overflow Incomplete is set, process as in 2 under Track Condition Check.

Track Overrun (byte 1, bit 1) -- There will be no error retry. The routine will set a return code of '04' in general register 15 and will return to the calling program.

Cylinder End (byte 1, bit 2) -- If the Overflow Incomplete bit is set, or if Read Variable Length Records is not set, or if the failing CCW is not the last in the original CCW list, then the processor will set on the Error flag and the Permanent Err flag, set DECECB to Complete With Error, set on the DECB ABEND Required flag bit, and set DCBIFL to indicate Permanent Error condition. The CSW is moved to the DECCSW, sense bytes 0 and 1 to DECSB0 and DECSB1, a message is provided to the operator indicating a permanent error and there is a return to the calling program with a return code of '0C' in general register 15.

If Overflow Incomplete is not set and Read Variable Length Records is set and the failing CCW is the last of the CCW list, then the routine will set DEBNIO to all 1 bits, set IORRV=0, set a return code of '08' in general register 15 and return to the calling program for Normal Completion processing.

File Protect (byte 1, bit 5) -- The command reject bit will also be set when this condition is detected. There will be no error retry. This is an ABEND situation.

MSAM Posting and Error Retry Routine (CZCMG)

This routine records the results of an I/O operation from or to a unit record device and determines future processing. MSAM Posting is called by the task monitor as the result of a synchronous I/O interruption following the execution of an IORCB by the MSAM Read/Write routine, or as the result of an asynchronous I/O interruption, such as when the operator has reset a jammed device. If the I/O operation was completed normally, MSAM Posting records it so that further processing may continue. If an error or unusual condition occurred, MSAM Posting records the condition and may determine whether to retry the operation or notify the operator before returning to the task monitor. (See Chart CC).

Attributes: Read-only, privileged, reentrant, nonrecursive.

Entry Points:
CZCMG1 -- Primary entry point from the task monitor with interruptions masked off. Type-1 linkage.

CZCMG2 -- Entry point following an asynchronous interruption. Type-1 linkage.

Input: Register 1 contains the address of the ICB.

Data References: CHAISA, CHADCB, CHADEB, CHADEC, CHASDT, CHASDA, CHAIOR, CHAICB, CHADBP.

Modules Called:
VMER (CZCRX) -- Informs the operator of the failing task I/O component and generates I/O error records.

VMSDR (CZCRY) -- Accumulates error statistics on task I/O devices.

SIR (CZCJS) -- Specify interruption routine.

DIR (CZCJD) -- Delete interruption routine.

Reset (CEAAH) -- Reset Device Suppression flag routine.

SYSER (CEAIS) -- System error processor.

WTO or WTOA (CZABQ) -- Write message to operator on console typewriter.

Exits:
Normal -- Register 15 contains 00.

Error -- Register 15 contains the return code passed from VMER or VMSDR.

Operation: What happens in MSAM POSTING depends on the type of device (reader, punch, or printer) on which the I/O operation occurred, and on whether the device is central (located at the central installation) or remote (located away from the central installation; remote job entry). Although the general logic in MSAM Posting is similar for all unit record devices, the sequence of operation and special considerations vary and require separate explanation. This description is divided into two parts. The first part explains MSAM Posting processing for central installation devices. The second part explains MSAM Posting processing for remote job entry devices.

CENTRAL INSTALLATION DEVICES

NORMAL I/O COMPLETION: On entry to MSAM Posting, the number of outstanding IORCBs is decremented by one. After falling through a series of abnormal condition tests, the Retry in Progress flag is set off in the DECB, the DECB is marked Normal Completion, the remaining DECBs are checked for Posting Reissue flag on (if so, the associated IORCBs are reissued), and return is made to the task monitor.

OTHER THAN NORMAL I/O COMPLETION: In general, where some abnormal condition is discovered, retry procedures are initiated, depending upon the condition and the device. If recovery is possible and the number of retries for a given condition has not been exceeded, the IORCB is reissued by MSAM Posting. If no recovery is possible, or if all recovery procedures have failed, the DECB is marked complete with error, and flags are set for unrecoverable and, if applicable, permanent error. A modified form of the CSW and the ISA sense byte are moved to the DECB, a message is sent to the operator with the WTO instruction, all DECBs whose Posting Reissue flags are set on are marked intercepted, the Posting Reissue flag is reset, and control is returned to the task monitor.

Priority of Checking I/O Results: The tests in MSAM Posting to determine results of the I/O operation are made in the following order for a card reader or card punch:

1. No Path Available
2. Purged I/O Operation
3. CCWs Not Relocated (specification error)
4. IORCB Intercepted
5. Start I/O Failure
6. CSW Status Zero
7. Prior Error Check (on normal completion)
8. Channel Control Check
9. Interface Control Check
10. Channel Data Check
11. Invalid CSW Status Bits Set
12. Unit Check - Sense Failure (invalid sense information)
13. Unit Check - Intervention Required
14. Unit Check - Command Reject
15. Unit Check - Bus Out Check (initial selection)
16. Unit Check - Bus Out Check (data transfer)
17. Unit Check - Equipment Check
18. Unit Check - Data Check
19. Unit Check - Unusual Command Sequence
20. Prior Error Check (not normal completion)
21. Program Check
22. Protection Check
23. Unit Exception

The priority for checking an I/O operation for a printer is:

1 - 12. Same as for reader and punch
13. Unit Check - Equipment Check
14. Unit Check - Code Generation Storage Parity
15. Unit Check - Intervention Required
16. Unit Check - Bus Out Check (data transfer)
17. Unit Check - Bus Out Check (initial selection)
18. Unit Check - Channel 9
19. Unit Check - Command Reject
20. Unit Check - Data Check (UCS option)
21. Prior Error Check (not normal completion)
22. Program Check
23. Protection Check
24. Unit Exception

Description of Posting and Recovery Efforts: Posting and recovery efforts in the order in which they occur (based on the priority lists above) are described below.

Name references appearing in parentheses at the end of paragraphs in the following descriptions correspond to block labels in flowchart CC and statement names (labels) in the listing of module CZCMG (MSAM POSTING).

No Path Available is set because no path is available and may be due to setting all paths disabled during alternate path retry. The associated entry in the SDAT for the device will be marked "phase out" by setting SDACE. The SDA entry must be locked during this change. A message is sent to the operator and permanent error posting is done (TESTLOCK).

A Purged I/O Operation indicated by IORPG causes an unrecoverable error to be set and return is made to the task monitor. A permanent error will be set if purged I/O occurs recursively more than ten times (TSTPURGE-UNRERR2).

If the CCWs are not relocated (specification error), an unrecoverable error is posted and a minor software.error is recorded by issuing the SYSER macro instruction (CCWSPEC).

IORCB Intercepted occurs when one or more IORCBs to be executed during an operation failed to be executed because of an interruption during an IORCB being executed earlier in the operation. .If no prior unrecoverable error has been recorded and retry is not already in progress, the IORCB is reissued (INCEPTED - PRIERR).

Start I/O Failure indication occurs for busy or not operational conditions. An alternate path is requested by setting IORAL, and a message is sent to the operator after requesting alternate path retry. If successful, normal posting occurs. If

unsuccessful, each alternate path is tried once and a message is sent to the operator. When all paths have been tried unsuccessfully, permanent error posting occurs (INCEPTED).

A Zero CSW should not occur and causes an unrecoverable error to be posted and a minor software SYSER (TSTCSW).

Prior Error Check indicated when CSW status bytes equal normal completion means some previous data may have been lost. The Error Retry flag is set off and an unrecoverable error is posted unless the I/O operation involves Form-D printing (that is, a dump). In that case, the operation is not considered unrecoverable and normal completion posting occurs (TSTSTAT).

Channel Control Check is one of four error conditions that is retried and then posted as an unrecoverable error when successful. This is because a record may have been lost or duplicated. These error conditions (channel control check, interface control check, invalid status or sense condition) set the error check flag, IOREC, to permit correct posting after retry occurs. If none of these error types occur after retry, then an unrecoverable error is posted. An exception is Form-D printing which is analyzed for a lower priority error or posted normal. After determining a channel control check condition exists, the flags IORAL and IOREC are checked to see if a prior error condition caused alternate path retry to be requested or if a prior Error Check condition occurred. If either flag is set, VMER is called to record a hard inboard error, the Alternate Path Retry flag is set, and a message is sent to the operator prior to reissuing the IORCB. If a prior error condition does not exist, the Error Check flag is set and the IORCB is reissued along the same path. A successful retry results in an unrecoverable error being posted (unless Form-D printing is underway). Unsuccessful retries result in one retry at each alternate path, with VMER being called and a message sent to the operator in each case (TESTCCC-ALTPATH1).

Interface control check occurs if there are channel or control unit problems. Processing is the same as for channel control check above (TESTICC).

Channel Data Check occurs when the channel detects a parity error in the information transferred to or from main storage on an I/O operation. VMER is called to record the error on each occurrence. Processing is as follows: Flags IORAL and IOREC are checked to determine if a prior error condition caused alternate path retry to be requested or a prior Error Check condition occurred. If either flag is set, alternate path retry is requested, VMER is called to record the error, and a message sent to the operator. If neither flag is set, the same path is retried according to the threshold values SDTCR0, SDTPU0, or SDTPR0 for the card reader, punch or printer respectively. Normal posting is done for successful retry. An alternate path is requested if the threshold value has been exceeded. A message is sent to the operator after requesting retry at an alternate path (TESTCDC-ALTPATH2).

Invalid CSW Status Bits cause processing similar to that for channel control check. The invalid bits are 32 (attention), 33 (status modifier), 34 (control unit end), 47 (chaining check), and 41 (incorrect length); these bits should not be set for unit record devices. VMER is called to record the error on each occurrance; a message is sent to the operator after requesting retry at an alternate path (TESTINVL).

Unit Check - Sense Failure is processed the same way as channel control check. A sense failure is caused by a sense command failure or invalid sense information. Sense bits 5 and 7 should not be set for the card reader or card punch. Sense bit 6 should not be set for the printer. Sense bits 4 and 5 may be set for the printer only if UCS is specified. All conditions can occur for either channel or control unit problems. VMSDR is called to record the error and a message is sent to the operator after requesting retry at an alternate path. Each alternate path is tried once (TESTUC).

The following unit check processing is dependent on the type of unit record device.

- For the reader or punch:

Unit check - intervention required occurs when the unit is not ready due to any of several conditions. Cards are not at each station (not EOF for reader), a stacker is full, the hopper is empty (not EOF for reader), the stop key is depressed the chip box is full or removed, or a card is jammed. A message to the operator is set up telling him that intervention is required. Asynchronous interruption proce dures follow. The message previously set up is sent to the operator (WTO). The Spe cify Interruption Routine (SIR) macro instruction is issued, the Error Recovery in Progress flag set on in the DEB, and control is returned to the task monitor. The operator performs the required action, correcting the condition requiring intervention or replacing the card on the equip ment or data check, and makes the device ready. The asynchronous interruption occurs, control is transferred to the Post

ing entry point, CZCMG2, the virtual storage IORCB is reissued, the number of outstanding IORCBs is incremented by one and return is made to the task monitor (INITSEL-ISINTV).

A Command Reject occurs when a command is given which the device is unable to execute. Posting is done for an unrecoverable error and if the DCB did not specify FORTRAN (formerly ASA) or machine control characters a minor software SYSER is recorded (TESTCREJ).

Unit Check - Bus Out for the reader or punch occurs when a parity error is detected on a bus out during either initial selection or command selection. Original path retry will be attempted the number of times specified in the SDT. A flag will be set in the DEB to indicate error recovery is in progress and the Error Retry Attempted flag set in the IORCB. The virtual storage IORCB will be reissued and control returned to the task monitor. If all retries on one path fail, VMSDR will be called to record hard outboard error statistics, a message will be written to the operator, and the IORCB will be reissued on an alternate path (BUSOUT-ALTPATH5).

A Unit Check - Equipment Check will cause retries on the same path to be attempted the number of times specified separately for reader and punch in the SDT. In each case, prior to returning to the task monitor, the Error Recovery in Progress flag is set in the DEB, the Error Retry Attempted flag is set in the IORCB, a message is sent to the operator, and an interruption routine is specified with the SIR macro instruction. The message sent to the operator directs him to take appropriate corrective action on the reader or punch. Such action creates an asynchronous interruption and MSAM Posting is entered at CZCMG2 where the Error Retry Attempted flag is set in the IORCB, the virtual storage IORCB reissued, and the number of outstanding IORCBs incremented by one before returning to the task monitor.

Where the maximum number of retries have been done, an alternate path is tried; a message is written to the operator requesting an alternate path, and again MSAM Posting returns until entered as the result of operator action. In the event alternate path retry is performed, a hard outboard error is recorded via VMSDR for the failing path (ALTPATH3).

Unit Check - Data Check occurs when an invalid card code is detected. This can occur only on a read command. If the card is a control card, two flags are set on in the DEB to indicate that a control card has been read. If the card is not a control card and the retry option in the DCB specifies no retries, the pocket option in the DCB is tested. This determines whether the Feeder and Stacker Select command is changed to use the stacker specified in the pocket option or whether the card is stacked as if no error occurred. A flag is set in the DEB to indicate that an invalid card has been read and accepted, a flag is set in the virtual storage IORCB to indicate error retry attempted, and the DEB Error Recovery in Progress flag is set on. The virtual storage IORCB is reissued at the next feed, select stacker command with command chaining suppressed, the number of outstanding IORCBs is incremented by one, and control is returned to the task monitor (RPBIT4-TESTPOCK-RETRY2).

If the card is not a control card and the retry option in the DCB specifies an unlimited number of read retries, a message is issued (WTO) informing the operator to replace the erroneous card for the retry attempt. Asynchronous procedures are handled as in Intervention Required, except that a flag is set on in the IORCB to indicate an error retry was attempted (CHKCOUNT-SETASYNC).

Unit Check - Unusual Command Sequence is caused by a read following a read with no intervening feed. This condition is allowable only if an error retry is in progress. Processing continues with the next CCW. If unusual command sequence occurs when an error retry is not in progress, posting for an unrecoverable error occurs (RPBIT6).

A Prior Error Check indicated when status bytes showed other than normal completion is treated as an unrecoverable error for a reader or punch (TESTEC - UNRERR2).

If a Program Check or Protection Check occurs on the reader or punch, there is no recovery procedure. Unrecoverable error posting is done and, if a protection check, SYSER is invoked to record a minor software failure (TESTPGC - TESTPTC - UNRERR2 - MISERR4).

A Unit Exception on the punch should never occur. If it does, permanent error posting is done and a minor software SYSER recorded. A unit exception on the reader indicates end of file (data set) and normal completion posting is done. A flag is set on in the DEB to indicate end of file (TESTPTC).

- For the printer:

A check is made for invalid sense bits as explained above under Unit Check - Sense Failure (PRUNITCK).

Unit Check - Equipment Check is processed similar to equipment check processing for the reader and punch. This unit check can be caused for a printer by either a hammer check or a buffer parity error (PRBIT3).

Unit Check - Code Generator Parity Error causes original path retry to be done the number of times specified in the SDT. It occurs only if the code generator storage is being reloaded for a printer with the universal character set (UCS) feature. If retry over the same path is unsuccessful, an alternate path is requested and a message sent to the operator. The error is recorded as intermittent when successful or as solid when unsuccessful for a path. The error is processed the same for each path, except each alternate path is tried once (PRBIT5).

If Unit Check - Intervention Required is the error, and the DCB for type is D or S (that is, form-sensitive), the operator will be sent a message to ready the printer. The SIR macro instruction will be issued to handle the expected asynchronous interruption. The error recovery in progress flag will be set on in the DEB and control will be returned to the task monitor. When the operator signals correction of the error condition by hitting the stop button followed by the start button, the asynchronous interruption will occur. Posting will be entered at its second entry point (CZCMG2), which was specified in the Specify Asynchronous Entry Condition macro instruction ICB. The virtual storage IORCB will be reissued beginning with the failing CCW, the number of outstanding IORCBs incremented by one, and control returned to the calling program. For form type-F, a message is sent to the operator directing him to mark the error page. The fixed area of the IORCB is moved from the interruption storage area to the DEB page. The address of the ICB in the DEB page is stored in the DCB, and the V- and R-con for posting entry 2 are stored in the ICB. The SIR macro instruction is issued to service the expected asynchronous interruption, the error recovery in progress flag is set on in the DEB, and control is returned to the task monitor. After the asynchronous interruption occurs, processing is identical to that done for error recovery on form type F for the printer after an equipment or data check (INITSEL - PRBIT1 - PTRINT - SETASYN2 - SETASYNC).

Unit Check - Bus Out is caused by a parity error on a command (initial selection) or data (data transfer) byte. Each path is retried the number of times specified in the SDT; when the maximum is reached, the failing path is recorded via VMSDR, a message is sent to the operator about the failure, and the IORCB is reissued along a new path (PRBIT2 - PRBOUT - RTRYBOUT).

If Unit Check - Channel 9 was sensed by the printer during the previous carriage motion and no other status or sense bits are set, processing continues with the next CCW. If FORTRAN (formerly ASA) or machine control characters have not been specified and there was a unit exception, the failing CCW is changed to a skip to channel 1 and processing continues. If any other sense bits are on, no special processing is done for the channel 9 indication (PRBIT7 - UEPRINT - NEXTLINE - RETRY3).

Unit Check - Command Reject is processed for the printer as explained for the reader and punch above (PRBIT0 - CMDREJ).

Unit Check - Data Check occurs only with a printer having the UCS feature when a code in data storage finds no match with any code in code generator storage. If SETUR had not been previously called by the user to load the buffer, a message is sent to the operator and an unrecoverable error is posted. Otherwise, the first time a data check occurs the buffer will be reloaded prior to the write retry. When the buffer has been successfully reloaded, the print retry counter is set to a number specified in the SDT and the failing CCW is changed to print without skipping, spacing, or command chaining. A flag is set in the IORCB to indicate error retry attempted, the IORCB is reissued, the number of outstanding IORCBs incremented by 1, and control returned to the task monitor. After the maximum number of retries have occurred, a message is sent to the operator indicating task ID and buffer arrangement. SIR macro instruction is issued to handle the possible asynchronous interruption from the operator. Posting is done for an unrecoverable error and control is returned to the task monitor.

If the operator wishes to accept the error and block further data checks, he will depress the stop and start buttons, thereby causing the asynchronous interruption. Posting invoked at its second entry point, CZCMG2, will proceed with the next line to be printed, except that the first CCW to be issued will be a block data command (UNRERR-RLDBFR-RETRY4-TSTCNT-ASKOPER).

A Prior Error Check (not normal completion) is treated as an unrecoverable error unless Form-D printing (that is, a dump) is in progress. If Form-D printing is in progress, MSAM Posting assumes the user is willing to have the job continue; processing continues to test for program check, protection check, or unit exception (TESTEC - UNRERR2).

If a Program Check or a Protection Check on the printer occurs, there is no recovery procedure. Unrecoverable error posting is done and unless FORTRAN (formerly ASA) or machine characters were not specified on the program check, SYSER is invoked to record a minor software failure (TESTPGC - TESTPTC - UNRERR2 - MISERR4).

A Unit Exception on the printer indicates the end of a page has been reached and a skip to the next page is required. If FORTRAN (formerly ASA) or machine control characters are in use (meaning the user handles his own skipping), the current IORCB will be reissued at the next CCW. The number of outstanding IORCBs is incremented by one and control returned to the task monitor. If FORTRAN or machine control characters are not in use, a skip to channel 1 command will be inserted over the last executed CCW, the virtual storage IORCB will be reissued beginning with the skip to channel 1, the number of outstanding IORCBs is incremented by one and control is returned to the task monitor (UEPRINT - NEXTLINE - RETRY3).

REMOTE JOB ENTRY DEVICES

For remote job entry devices (reader or printer; a punch is not supported), normal I/O completion results in processing identical to that for central installation devices plus several additional processes. Buffer input received from a reader is checked for the presence of an ETX character; where found, it indicates end-of-data-set (end-of-file) and a flag is set. Where unissued IORCBs are to be reissued as the result of a prior unsuccessful completion, synchronization of ACK characters is assured for the reader before an IOCAL is issued (TESTRJE - SETENOF - REISS1 - CHECKACK).

In general, MSAM Posting performs only posting functions for remote job entry devices. Necessary error recovery procedures are performed by the resident supervisor. MSAM Posting error processing for both the remote reader and printer is explained below.

Channel control check, interface control check, and channel data check errors are recorded via VMER, CHKINTM is called if intermittent errors are present, and a message is written via WTO to the TSS operator informing him of the permanent channel error (NORMRJE - TESTICC - TSTCDC).

Unit check - sense failed error is recorded via VMSDR, permanent error is posted, CHKINTM and WTO are called (TSTDUC).

Unit check - time out error is recorded via VMSDR, permanent error is posted, CHKINTM and WTO are called if it is a 'should not occur' error (IORJESN is on). If it is a 'should occur' error (IORJESN is off), then the operation is set up to be handled by an asynchronous interruption (TOCHK).

Unit check - intervention required results in the first byte of the DECB being set to hex 14 so that BULKIO may reinitialize the job (IRCHK).

Unit check - lost data, data check, or overrun errors are recorded via VMSDR, permanent error is posted, CHKINTM and WTO are called. If IORJESN is on, subsection 2 of the error counter tables is used for recording the error. If it is off, subsection 1 is used (DATACHK).

Unit check - busout, equipment check, or command reject errors are recorded via VMSDR, permanent error posting is done, CHKINTM and WTO are called (BOCHK - EQUIPCHK - COMMREJ).

If no sense bits were found on, a minor software SYSER is called.

Unit exception - first, intermittent errors are recorded using the CHKINTM subroutine. Then, if receive mode, the first byte of the buffer is checked for an EOT. (An EOT will be found if the operation was completed in the previous buffer but there was no room to write the EOT, or if a card jam occurred just as the buffer was being completed.) If there is an EOT, the DCBENOF flag is checked, and if on normal completion is posted. If off, it indicates either a card jam or the 'STOP' button was pushed. The operation will then be set up to be handled by an asynchronous interruption (SKPINTM - TSTEOF - CHKSAIN).

If the first byte of the buffer does not contain an EOT, then the data buffer being read at the time of the interruption is checked for ETX or ETB. This check will be made at the middle of the buffer (byte 83) if IORJEOC is on, or at the end of the buffer (byte 167) if IORJEOC is off. If an ETX, the operation has completed normally, DCBENOF is set on, and normal completion is posted. If ETB is found, either a card jam has occurred or the STOP button has been pushed. Setup will be made for an asynchronous interruption. If neither ETX or ETB is found, a minor software SYSER is posted (SETEOF - CHKSAIN - RJEMSER1).

For unit exception transmit mode, incorrect length is checked. If off, normal completion is posted. If on, either a paper jam exists or the printer was stopped by pushing the stop button. In either case

the operation is set up for an asynchronous interruption (CHKFORM).

Incorrect length errors are recorded via VMER, any intermittent errors are recorded and a message is written to the operator via WTO informing him of the permanent error (CHKIL).

Control unit end, program check, chaining check, and protection check errors are posted as permanent errors, intermittents are recorded, and a message is written to the operator via WTO (CUEND - CHAINCHK).

Attention, status modifier, and busy errors are handled the same as incorrect length (D9) (BUSYCHK - ATTNCHK - STATMOD).

If none of the status bits are found on a minor software SYSER is called (RJEMSER2).

TAM Posting Routine (CZCZA)

After the termination of a TAM Read/Write initiated I/O operation, control is passed by the task monitor to TAM Posting to process this I/O interruption. TAM Posting analyzes the interruption data to determine the action to be taken. It also examines the input message content to determine the buffering technique to use and if user errors have occurred. When errors occur, both recovered and unrecoverable error data is recorded. (See Chart CD.)

Functions provided by TAM Posting are:

- Posting of completed I/O actions.
- Translation and movement of user data on read operations.
- Continuation of TAM Read/Write operations which involve multiple IORCB generation.
- Posting of attention signaling during input or output operations while the terminal is transmitting or receiving.
- Detection of errors or exceptions terminating the channel program.
- Decoding of errors or exceptions and initiating possible recovery action.
- Posting of error or exception data when recovery has not been requested or the error is nonrecoverable.

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged.

Entry Point: CZCZA1 -- Entered via type-1 linkage.

Input: The IORCB and the ISA contain information relative to the condition under which the I/O operation was terminated.

Data Reference: CHAISA, CHAIOR, CHADEC, CHATOS, CHADCB, CHADEB, CHASDA.

Modules Called:
GETBUF (CZCMA) -- Get a buffer area.

WRITE (CZCYM) -- TAM Read/Write.

WTO (CZABQ -- Write to operator.

RESET (CEAAH) -- Permit task to access I/O device.

IOCAL (CEAAO) -- I/O call.

VMER (CZCRX) -- Virtual memory error recording.

VMSDR (CZCRY) -- Virtual memory statistical data recording.

SYSER (CEAIS) -- System error.

Exits:
Normal -- Return to the task monitor.

Error -- SYSER is called in the case of an undefined interruption, no SDAT entry, an undefined inboard error, or no sense data.

Operation: TAM Posting is called by the supervisor as the result of an I/O completed interruption. It saves the general registers and then initializes them with pointers to the IDA, DECB, DCB, TOS, and IORCB buffer and CCW list.

TAM Posting then decodes the reason for the interruption of the I/O operation (see Figure 7). On determining that the interruption does not reflect a normal completion, a branch is made within TAM Posting to initiate the appropriate recovery procedure. If this is not possible, errors or exceptions are analyzed, and flags are set.

The sequence to determine the interruption type follows:

- IORCB flags - The flags in the IORCB are checked first to determine if there was a HALT I/O or a START I/O failure. An exception condition detected by IOS is indicated by setting flags in the IORCB.
- Inboard failures - The inboard (channel) type of failures are tested first as a group (incorrect length, program check, protection check, channel data check, interface control check, chaining check). If any one is found set, control is transferred to the inboard failure analyzer.

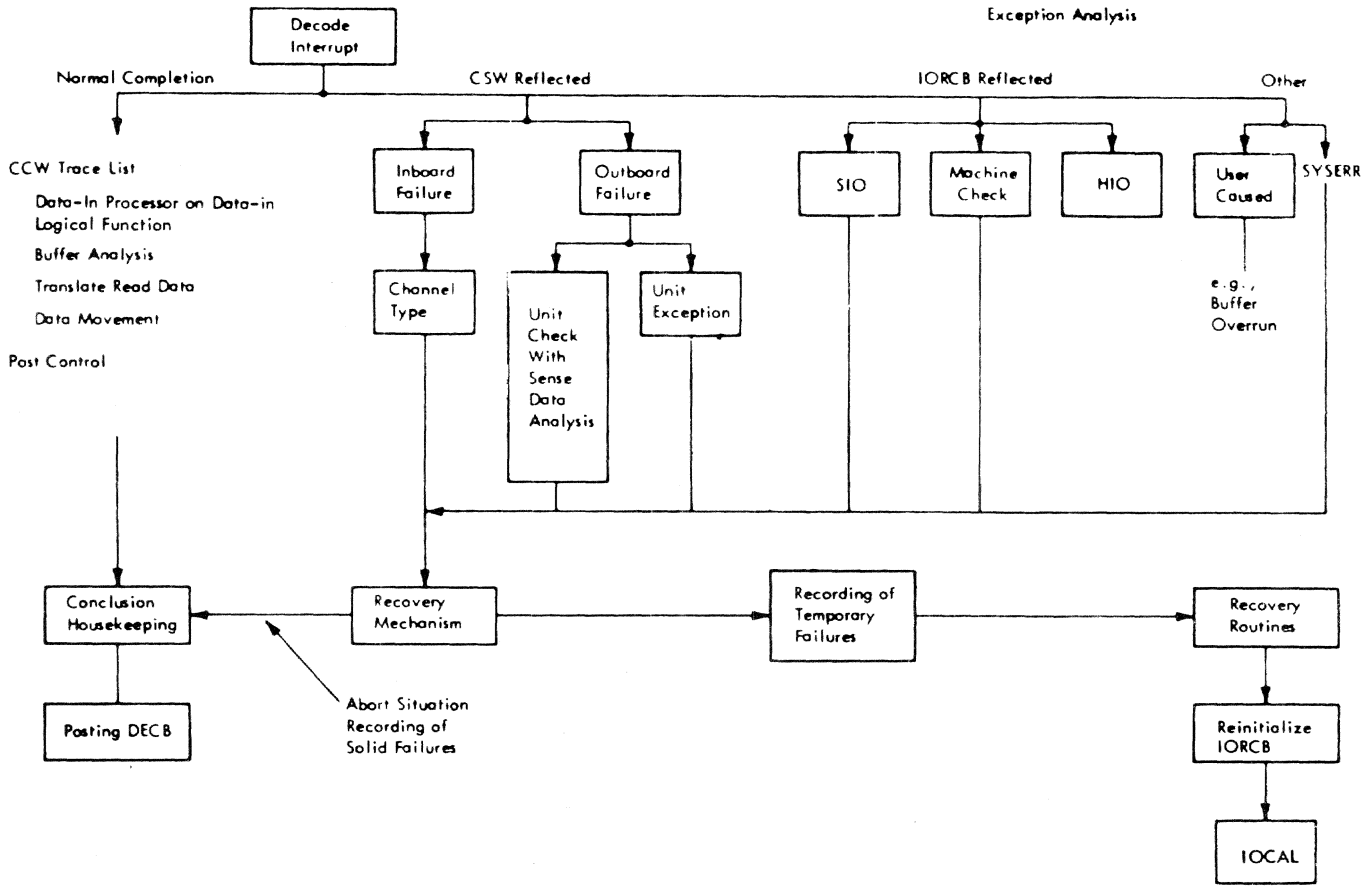


Figure 7. TAM Posting: Normal Completion and Exception Analysis Paths

- Outboard check or exception - The outboard failures (unit check and unit exception) are tested individually. If either is set, control is given to the appropriate outboard failure analyzer.

- Normal completion - The channel end (CE), device end (DE), and program controlled interruption (PCI) bits are tested. The detection of any CE, DE, or PCI bit being set will transfer control directly to the CCW trace list.

- Undefined operation - If there was no detected error or no normal condition, TAM Posting identifies this I/O interruption as an undefined operation and a SYSER is declared.

The CCW trace list is then traced directly when normal completion was decoded from the interruption. TAM Posting performs this CCW trace by processing each CCW in the CCW list. This CCW list was just executed previously by IOS and is returned by IOS in the IORCB area of the ISA. For dynamic buffering the length is obtained from TOS and is indicated in Table 17.

Table 17. TAM Posting: Terminal Length Statistics

Terminal Type	Standard Length	Maximum Single Input Record Buffer
IBM 1050	130	260
IBM 2741	130	260
IBM 1052-7	130	260
TTY MOD 35	72	144

TAM Posting uses the logical function list kept in TOS associated with this IORCB for determining necessary processing requirements for each CCW. in logical function list contains a one byte code for each CCW in the channel program. For each CCW and its associated logical function code, the IORCB buffer pointer is stepped to the proper buffer location and the CCW list pointer is stepped to the next CCW. TAM Posting then processes the next logical function until the entire list has been

processed. Processing then continues with post control. If the Data-In logical function code is present it performs the previous action and also branches to the Data-In processor. This Data-In assures that a user buffer area is available, and translates and moves data to this buffer area when required. It is the initial Data-In CCW, in the CCW trace list, that determines the user buffer area requirements.

The Data-In processor is entered if there is a Data-In logical function in the CCW list. The Data-In processor coding determines input buffer length, detemmines input buffer processing requirements, translates input data when required, moves data to user buffer area, and maintains input data bookkeeping.

The user may supply the necessary buffer area, by specifying in the TAM Read/Write macro instruction, the address and length of this area. The user may also specify, in other parameters in this DECB or in the DCB, that dynamic buffering is requested. (See Table 18.)

If TAM Posting issues the GETBUF macro instruction to obtain dynamic buffer areas, the user must return these buffer areas, when processing of the input data is complete.

The input buffer length and processing requirements are determined by doing a translate and test of the input data. The data in the translate and test table (ZATDIT) contains only one functional character, at present, which senses the end of the input data record. The ZATDIT functional character code is used to do a table lookup for the processing routine to be called. The data in the processing control table (ZATDIC) is provided for this purpose and contains the location of the end-of-record processor. This end-of-record processor determines if translation of the input data is required and translates the data to EBCDIC when necessary. The translation table address is obtained from TOS. If the field is zero, no translation is required.

One error condition may be determined as a result of the translate and test. This is an input buffer overflow and exists when the terminal operator inputs a single record length greater than the specified length for the terminal. This condition will cause the Master Exception flag in the Posting flag field of TOS to be set, and the Input Buffer Overflow flag bit to be set in the flag field of the DECB. When the translation requirements are completed the message is checked for the end of line (EOL) sequence of the input record (illustrated in Table 19).

It is determined that the terminal has a specified EOL sequence when the EOL sequence count in TOS is other than zero.

Table 18. TAM Posting: Specification of User Buffer

DCB	DECB	Comments
Option ignored	L=C	Conversational move, dynamic buffering length is twice standard terminal length.
Option ignored	A=S L=N	Dynamic buffering. Length is stated value=N.
L=N	A=S L=S	Dynamic buffering. Length is stated value=N (from DCB).
BUFTEK=DYN L=N	L=S	Dynamic buffering (from DCB). Length is stated value=N (from DCB).
BUFTEK=DYN Length ignored	L=N	Dynamic buffering (from DCB). Length is stated value=N.
Option ignored	A=address L=N	Buffer address is indicated. Length is stated value=N.

L - LENGTH

L=S - implies length in DCB.

Dynamic Buffering - Buffer obtained by POSTING and address passed to user in DECB. A user is therefore responsible for returning these dynamic buffers by using a FREEBUF.

Table 19. TAM Posting: Expected EOL Sequence

Terminal	Sequence
IBM 1050	CR, B
IBM 2741	CR, C
IBM 1052-7	NONE
TTY MOD 35	CR, L.F., XOFF

CR - Carriage Return

B - End of Block

C - End of Transmission

L.F. - Line Feed

XOFF - Transmitter Off

When a successful comparison cannot be made the device type code is obtained from the TOS and used to do a table lookup for the abnormal EOL routine for that device.

The IBM 1050 and 2741 use a common Abnormal End of Line routine and it is entered whenever the ending sequence is other than the expected. The routine will determine one of four possible conditions:

1. B--EOB character only -- Normal allowable ending which may indicate card input of inter-line record formatting. Detection of this condition will inhibit the placing of the New Line code character in the data record and will return to the normal program flow.

2. Last character received is C--EOT -- If this character is received the Unit Exception flag in the Post Flag field in the CHATOS is tested, and if set, the Message Complete flag will be set and the return is to normal Data-In processing flow. If the Unit Exception flag is not set, the Attention on Read and Master Exception flag are set in the Posting flag field of the CHATOS and the Attention flag bit in the Pre-Post Data field in the CHATOS is set. Control is then returned to the normal Data-In processor flow.

3. Neither B nor C -- The Master Exception flag in the Posting flag field of CHATOS is tested. If on, control is returned to the Input Data processor to allow the record in error to be moved to the user buffer area.

4. If the Master Exception flag is not set, an undefined system error has occurred and control is given to the undefined error routine. Should the record length prove to be greater than the user the user buffer remaining count field, that portion of the record equal to the user buffer remaining count is moved to the user buffer areas, the User Buffer Overflow flag in the flag field of the DECB is set, and the master exception flag bit in the posting flag field of the TOS is set. Normal processing continues with control being returned to the CCW trace list as if no user buffer overflow had occurred.

If the EOL sequence comparison is successful the first character of the EOL sequence is overlaid with an EBCDIC new line (NL) character and the record length is adjusted to include the text plus the NL character. This record length is then compared to the user buffer remaining count field of TOS. If it is equal or low, the record is moved from the input buffer to the user buffer area. The user buffer pointer, user buffer remaining count, and input character count fields in the TOS are updated. Control is then returned to the CCW trace list.

The CCW trace list coding continues for each logical function byte until the entire CCW list has been processed. When the last CCW is processed, we continue processing with the proper post control coding as determined by the option type-code reference in the post control table.

The functions of the post control routine coding are to determine the proper posting and control for:

- Read operation entry
- Write operation entry
- Write with response entry
- Control order entry

If a read request is complete within this IORCB, the area and length are moved to the DECB and common DECB posting continues. If the read is not complete, TAM Read/Write is called.

If a write is complete within this IORCB, common DECB posting continues. If the write is not complete, TAM Read/Write is entered as a subroutine.

If the write with response is complete within this IORCB, the data in the buffer bit in the Posting flag field of TOS is tested. If set, the Message Complete flag is set in the Post flag field and control is transferred to post control read. If the data in buffer bit is not set, enter TAM Read/Write is entered as a subroutine.

If write with response is not complete, TAM Read/Write is entered as a subroutine.

If a control order entry, common DECB posting continues. Common DECB posting ends post control by making an initial test of the Master Exception flag in the Posting flag field of TOS. If set, the ECB field of DECB is posted complete with error and the synad request is set in the DECB flag field. If the Master Exception flag is not set, the ECB field of DECB is posted complete without error and the prepost data field is moved to the DECS flag field.

Common DECB posting then moves CSW and sense data from ISA to DECB. If the Master Exception flag is set, TAM Posting does not allow entering TAM Read/Write as a subroutine but continues with common DECB posting.

After completing common DECB posting the housekeeping completion routine coding is executed. The housekeeping function includes:

- Recording unrecoverable error data
- Recording recovered error data
- Clearing error counters and error data fields
- Preparing line on abnormal end and return
- Issuing status of IORCB and return
- Setting up normal return to task monitor.

Note that TAM Posting does not issue an ABEND. On determining any exception or error condition, the appropriate information is posted in the DECB and at the proper time a return is made to Task Monitor.

Only if TAM Posting cannot determine a path to follow will it issue a SYSER.

TAM Posting provides exception analysis error decoding and recovery actions. The processing of all exceptions except attention interruptions posts all data to the user, whenever error recovery is not indicated or possible. The following flags may be set in TOS to communicate unrecoverable termination status to the user via the DECB:

- Unit Exception flag - set when a function is terminated by unit exception.
- Master Exception flag - set whenever an exception interruption has terminated the action.
- Abort flag - set whenever maximum error recovery attempts have been unsuccessful.
- Attention flag - set whenever attention signalling is detected during a read type operation.
- Recovery in Progress flag - set whenever an error recovery action is initiated.

An example of Start I/O (SIO) failure follows. The SIO Failure flag in the IORCB is interrogated by the Define Interruption routine. If the flag is found to be on, transfer is made to the SIO-HIO failure processor.

The status in the CSW is then interrogated to see if a unit check or unit exception has occurred. If either bit is on, control is turned over to the proper processor. If neither bit is on, a message is sent to the system operator informing him of the Start I/O failure and the associated symbolic device.

The ABEND Required and SYNAD Requested flags are set in the flag bytes of the DECB. The ECB field of the DECB is set to indicate "complete with error". A branch to the Housekeep and Return subroutine occurs.

The two types of outboard failures are unit check and unit exception. Each has its own control routine coding whose purpose is to determine the recovery action to be initiated. Each of the recovery routines is coded to operate for unique combinations of conditions.

As an example of error recovery on unit check, TAM Posting starts with the interrupted CCW obtained through the data provided in the ISA and the IORCB located in the ISA. The sense byte bits in the ISA are then tested in a predetermined priority.

When a test is found to be positive, a counter located in TOS which represents that sense condition is stepped and tested for a maximum count. Refer to Table 20 for CSW status and sense data typical 'maximum exception retry' counts.

When any sense condition retry counter is stepped to maximum, the Abort flag is set to indicate an unrecoverable condition, the failures are recorded, and TAM Posting branches to conclusion housekeeping. If it is a recoverable condition the internal recording saves pertinent error data in TOS. After this recording, the Unit Check Director routine coding is entered, to access the Unit Check Recovery routine to be initiated as determined by the interruption conditions.

The Reinitialize IORCB subroutine is provided to be used by all unit check action routines. Prior to entering, all altering of the channel program and buffer data is completed. This routine initializes all relative addresses in the IORCB, sets the necessary flags, and issues the IOCAL macro instruction. TAM Posting then sets up for a return to task monitor.

The following describes typical unit check recovery routine coding provided:

Retry Full IORCB on System Error: This action is used where the channel program has been terminated in such a way as to require a retry of the full program or posting of a system error which is unrecoverable. The routine will first test the Abort flag in TOS. If set, it will set a

Table 20. TAM Posting: CSW Status and Sense Data Typical Maximum Exception Retry Counts (Extracted from CHASDT)

CSW Status	Bit No.	Retry Count
Attention	32	N.A.
Status Modifier	33	N.A.
Control Unit End	34	N.A.
Busy	35	N.A.
Channel End	36	N.A.
Device End	37	N.A.
Unit Check	38	3
Unit Exception	39	3
PCI	40	N.A.
Incorrect Length	41	3
Program Check	42	3
Protection Check	43	3
Channel Data Check	44	3
Channel Control	45	3
Interface Control	46	3
Chaining Check	47	3
Consecutive Errors		10
SENSE DATA		
Command Reject	0	3
Intervention Required	1	3
Bus Out	2	3
Equipment Check	3	3
Data Check	4	3
Overrun	5	3
Receiving	6	3
Time Out	7	3

System Error and ABEND Requested flag in the DECB and enter the CCW trace list routine. If the Abort flag is not set, the routine will set the Retry in Progress flag in TOS. The Reinitialize IORCB routine is then entered.

System Error Retry from Interrupted CCW: This action is used whenever a system error has occurred and it is necessary to restart the program from the interrupted CCW. The Abort flag in TOS is tested. If set, the routine will set an error condition in the DECB and enter the CCW Trace List routine. If abort is not set, the Retry in Progress flag will be set. The relative location of the interrupted CCW is saved and the Reinitialize IORCB routine is entered.

Record Only: This action is used when an error has occurred which will not effect channel program operation, but must be recorded as a system history. It also may be used when the terminal is not equipped with error correction. A branch occurs to the CCW Trace List routine.

Unit exception, for example, can be caused by a negative response from the 1050, either during addressing or polling.

IOREQ Posting Routine (CZCSE)

After a system interruption which occurs at the completion of an IOREQ initiated I/O operation, control is passed by the Task Monitor to IOREQ Posting to process this I/O interruption. The address of IOREQ Posting is specified by the posting address constants in the IORCB.

IOREQ Posting analyzes the interruption data to determine the action to be taken. It then posts the normal or abnormal completion code in the DECB, allowing the Check routine to later take action based on these codes. IOREQ Posting is executed in privileged mode, with all other task interruptions masked off. (See Chart CE.)

Attributes: Reentrant, resident in virtual storage, closed, nonrecursive, read-only, privileged.

Entry Point: CZCSE1 -- Entered via type-1 linkage.

Input: None. Information relative to the conditions under which the I/O operation was completed is in the IORCB and the ISA.

Data References: CHAISA, CHADEC, CHADEB, CHADCB, CHAIOR.

Module Called: Reset (CEAAH) -- Resets error flag so that task can access I/O device.

Exits:
Normal -- Return to calling program.

Error -- ABEND macro instruction.

Operation: IOREQ Posting initially saves the general registers. A check is made to verify that the IORCB has been executed (that is, the channel program has been attempted).

If the IORCB was not executed, a check of the Error flag for the corresponding DEB (pointed to by the IORCB) is made. If the error flag is set, an intercepted condition is set in the ECB (pointed to by the IORCB) of the DECB. The allowed IORCB count field in the DEB is decremented by 1 an a return to task monitor occurs. If the error flag is not set, an exit to ABEND occurs.

If the IORCB was executed, the CSW is moved to the DECB, the ISA sense information is moved to the DECB, the allowed IORCB count in the DEB is decremented by 1, and the input data in the IORCB data buffer area, if buffering is used, is moved to the data area in the user virtual storage pointed to by the DECB.

If START I/O failed 'Complete with Error' is posted, DECBSF is set to show SIO failure, SYNAD is requested, and the IORCB count decremented. The return code is set to zero and a return to the task monitor is effected.

The CSW bits are examined and, depending on their setting, normal or error completion is posted. If the CSW=0, ABEND is requested and error and permanent error flags set in the DEB. IOREQ Posting does not perform any error recovery.

For normal completion, "normal completion" is posted in the ECB of DECB, and a return to the task monitor occurs.

For abnormal completion, "complete with error" is posted in the ECB of DECB, a request is set for SYNAD in DECB, the error flag is set in DEB, and a return to the task monitor occurs.

Check Routine (CZCRC)

The Check routine is entered as the result of the CHECK macro instruction that a user issues to ensure the completion of a previous READ or WRITE macro instruction. To determine completion or other results, Check examines the DECB. SAM Posting and Error Retry will have posted information in the DECB if the I/O operation is complete. If not complete, Check awaits completion. On completion (successful or otherwise), normal return, an ABEND, or set-up for entry to a user's SYNAD routine occurs, depending on switches set by SAM Posting and Error Retry. Check also calls SAM Mainline EOV for necessary end-of-volume processing. If end of data set is encountered, Check sets up entry to a user's EODAD routine before returning to the caller.

Attributes: Reentrant, resident in virtual storage, closed, privileged, nonrecursive.

Entry Point: CZCRCS -- Entered via type-1M or type-2 linkage.

Input: Register 1 contains the address of the DECB to be checked.

Data References: CHADEC, CHADEB, CHASDA, CHADCB.

Modules Called:
SAM Mainline EOV (CZCXE) -- SAM EOV mainline processing.

SAM Read/Write (CZCRA) -- SAM Read/Write.

AWAIT (CEAP7) -- Await an interruption.

TWAIT (CEAR0) -- Terminal I/O wait.

Exits:
Normal -- Returns to the calling program with zero in register 15.

Other conditions --

- If a SYNAD condition, returns with:

  Register 0 - DECB address
  Register 1 - 8-bit SYNAD code
  Register 15 - Address of parameter list containing user's SYNAD V-con and R-con addresses.

- If an EODAD (end of data set) condition, returns with:

  Register 1 - DCB address
  Register 15 - Address of parameter list containing user's EODAD V-con and R-con addresses.

- ABEND occurs if EODAD or SYNAD exits needed but not provided. Requested by SAM Posting and Error Retry. I/O requests checked out of sequence.

Operation: If the In-Use flag in the DECB is not on, Check has been entered to check a DECB for which no I/O is outstanding. Therefore, control is immediately returned to the calling routine.

When the DECB is in use but the I/O operation is not complete, the AWAIT macro instruction is issued to wait for the expected I/O event to complete.

When the I/O event has occurred, the DECB is checked for errors. The task will be abnormally terminated when the SYNAD or EODAD exit is to be given control and it is not supplied.

Note: When an intercepted DECB is checked and it has an end of volume request posted in it, end-of-volume processing is performed as explained below. However, if the intercepted DECB has no end-of-volume request this means that the I/O associated with the DECB was never attempted. Therefore, Check links to the Read/Write routine to reissue the I/O request. Check waits until this request is complete before doing any other processing.

If complete with errors is posted, and if there is a unit check caused by reading backwards into a load point, the end of volume processing is performed as explained below.

If the DECB is marked complete with errors, and there was no read backward into load point, this means that an I/O error has occurred, and the system retry procedures cannot correct it. The Check routine first tests the ABEND bit (in the DECB), if

the data set is sequentially organized. If on, this means that the error is catastrophic, and the task cannot continue. ABEND is then called. If the ABEND bit is off, then the SYNAD request flag (in DECB) is tested, and the user's SYNAD exit is entered. If SYNAD is not on, a normal return is given to the user.

If the data set is not sequentially organized, the Check routine proceeds as described above for the case in which the ABEND bit is found off.

End-of-Volume Processing: Check calls the Mainline EOV routine which performs various end of volume processes and will do volume switching if necessary. When control returns to Check, if the end of volume corresponded to end of data set, the user's EODAD exit is set up (if supplied). When the end of volume condition does not correspond to end of data set, the read request DECB which caused the EOV request is restarted. Should it be a write request DECB which causes the EOV condition, a normal return is made to the calling routine with the DECB marked "Complete, No Errors".

Note that when conditions arise which require a branch to SYNAD or EODAD, the Check routine only sets a pointer to the R-con and V-con of that routine in register 15, and returns to the calling program. The point returned to will be within the expansion of the CHECK macro instruction. If general register 15 is zero, the next sequential instruction after the expansion should be given control. If non-zero, the succeeding instructions of the expansion must set up a type-1 linkage with the supplied R- and V-type address constants.

CLOSE PROCESSING

The following routines describe the CLOSE processors for SAM, MSAM, TAM, and IOR, as well as Close Common and MSAM Finish.

Close Common Routine (CZCLB)

The Close Common routine will logically disconnect the data set from the problem program, close the data control block, and relinquish main storage. It then branches to the appropriate access dependent close routine to complete the closing. (See Chart DA.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged routine, public.

Entry Point: CZCLBC -- Entered by type-1 or type-2 linkage.

Input: Register 1 contains the address of the CHAGSM table. The CHAGSM table is generated by the expansion of the CLOSE macro instruction and consists of one double word entry for each DCB (and its associated data set) to be closed.

Data References: CHAGSM, CHADCB, CHATDT, CHADEB, CHADHD.

Modules Called:
SAM Close (CZCWC) -- SAM close.

TAM Close (CZCYG) -- TAM close.

MSAM Close (CZCMI) -- MSAM close.

IOR Close (CZCSD) -- IOR close.

VAM Close (CZCOB) -- VAM close.

VMA (CZCGA) - Free virtual storage.

Exits:
Normal -- Return to the calling program.

Error - - ABEND macro instruction.

Operation: This routine performs close processing only if the DCB to be closed is open. If the close is temporary, the number of times the data set has been open is not decremented as is the case in a normal close.

Close Common then transfers control to the appropriate access dependent close routine. When the access dependent close routine returns, all storage assigned to the DCB is released via FREEMAIN, unless the close is temporary in which case the assigned storage is not released. The DCB is restored to its pre-open condition. If this is not the last DCB to be closed, CHAGSM points to the next DCB to be processed.

SAM Close Routine (CZCWC)

Called by Close Common, SAM Close positions the data set volume, releases storage allocated for the DEB, disconnects the DEB from the chain of DEBs, and returns unused DA extents to external storage (Chart DB).

Attributes: Reentrant, resides in virtual storage, closed, read-only, privileged.

Entry Point: CZCWC1 - Entered by type-1 linkage.

Input: Register 1 contains the address of the DCB. The SAM communication block (CHASCB) is defined in the PSECT of SAM close and contains three temporary control blocks - DCB, DEB, and DEC - which are used during the closing process to perform label processing.

Data References: CHASCB, CHADCB, CHADEB, CHATDT, CHASDA, CHADEC.

Modules Called:
Tape Positioning (CZCWP) -- Tape positioning.

User Prompter (CZCTJ) -- Communicate with user.

VOLCVT (CZCWV) -- Volume address convert.

VMA (CZCGA) -- Free virtual storage.

GIVBKS (CZCEG) -- Release unused SAM external storage.

Control (CZCRB) -- Magnetic tape positioning.

Mainline EOV (CZCXE) -- SAM mainline EOV processor.

SETDSCB (CZCXS) -- Set DSCB.

QSAM (CZCSA) -- To handle end-of-volume condition.

AWAIT (CEAP7) -- Await an interruption.

Exits:
Normal -- Return to calling routine.

Error -- Via ABEND.

Operation: If the device indicated in the DCB is direct access, the T-Close flag is turned off. DA data sets may not be temporarily closed.

The SAM control block (CHASCB) is initialized by zeroing out the variable portion, storing pointers to the DCB, DEB, and JFCB into it, and initializing the temporary DCB and DEB.

The QCLOSE subroutine is bypassed if neither GET nor PUT is indicated in the DCB. If the data set being closed has been processed by QSAM, QCLOSE, an in-line section of SAM Close, is entered to perform functions unique to the closing of a QSAM data set. Control is given to the TREOV section of QSAM, which checks for outstanding read requests and flushes any which exist, or writes out any buffer which has been partially or completely filled, but not yet written out. (See the QSAM section of this publication for a description of TREOV.) A FREEMAIN macro instruction is next issued to release the storage obtained for the work area and buffers by QOPEN. At this point, QCLOSE processing is complete and normal processing of SAM Close continues.

If the device is a magnetic tape drive, the common portion of the input DEB is copied into the common portion of the temporary DEB, and a pointer to the temporary DCB is stored into the temporary DEB. Fields in the temporary DCB are set so it may be used for label processing.

SAM Close uses the AWAIT macro instruction to make sure that all user I/O is complete before the data set's close processing is continued. Incomplete input operations will be purged by SAM Close.

Unused extents are released, via GIVBKS, if the following three conditions are met:

a. device is direct access

b. at least one full track is unused

c. release of unused extents is specified in the JFCB.

The Mainline EOV routine is then entered to complete end-of-volume processing if the last I/O operation was a write. When control is returned to SAM Close, and if the device is a magnetic tape drive, the tape volume is positioned as specified in the close options by use of the Control routine.

MSAM Finish Routine (CZCMH)

MSAM Finish is used to complete processing for a data group mounted on a unit record device. MSAM Finish can also be used to signal the end of the current data group without closing and reopening the DCB for the next data group. (Chart DC.)

Attributes: Privileged, reentrant, read-only, public, system, nonrecursive.

Entry Points:
CZCMH1 -- Primary entry point via type-1 or type-2 linkage.

CZCMH2 -- Asynchronous interruption entry point via type-1 linkage.

CZCMH3 -- Synchronous interruption entry point via type-1 linkage.

Input:
For entry at CZCMH1, register 1 contains the address of the DCB.

For entry at CZCMH2, register 1 contains the address of the ICB.

For entry at CZCMH3, the ISA contains the IORCB.

Data References: CHADCB, CHADEB, CHADEC, CHAIOR, CHAISA, CHADBP.

Modules Called:
WTO (CZABQ) -- Write message to operator.

SIR (CZCJS) -- Activate an interruption routine.

DIR (CZCJD) -- Delete an active interruption routine.

Reset (CEAAH) -- Reenable a device after I/O error.

Exits:
Normal -- For CZCMH1 return, register 15 contains one of the following return codes:

'00' Completed successfully.

'04' Incomplete.

'08' Complete with I/O error; if PUT, register 1 points to failing record, register 0 points to user DECB.

For CZCMH2 and CZCMH3 return, register 15 contains zero.

Error -- Abnormal termination via ABEND macro.

Operation: In order to avoid an automatic wait in MSAM Close, the FINISH macro should be issued and subsequently reissued by the user until its return code indicates completion. When the FINISH macro is issued by the user or by the MSAM Close, CZCMH1 is entered. Following an asynchronous interruption (caused by operator response to a message), CZCMH2 is entered; CZCMH3 is entered after a synchronous interruption (caused by completion of the card-read-and-stack operation).

The user can stop Finish processing by setting off the Finish in Progress bit in his DCB and calling Finish for the final time. This will cause any activated interruption routine to be deleted before a return code indicating "complete" is given to the user.

The user may also prevent the issuing of a data-group-end message by turning on the Inhibit Message bit in his DCB. This will cause Finish to go directly to its completion routine at points where it would otherwise issue a message and wait for response.

Primary Entry Point (CZCMH1) Processing: If the DCB or DEB are not valid, the task is terminated via ABEND.

When the input is complete without error, the message-defining loop is entered. (See DCBRCX=30, below.) If the input is incomplete, control returns to the user with general register 15 indicating "incomplete". If the input is complete with error, the completion routine (see DCBRCX=80 below) is entered.

When Finish is first entered on output, and an error was recorded by a user-issued PUT, no attempt will be made to flush the buffer. Otherwise, if no error occurred, the device type determines the line of processing.

For a printer, a PUT is issued to write the last buffer page. If the PUT is not yet complete, the return code is set to "incomplete" and the routine returns to the caller. If the PUT is complete but an error on a Finish-issued PUT is indicated, the user will be provided with error pointers if this has not already been done (error recovery will be attempted if the error indicator is set off), and control returns to the caller with the Finish-Just-Issued flag on, the Finish-in-Progress flag off, and a return code indicating "complete with error" in general register 15.

Otherwise, the message "Remove output from printer XXXX, then ready printer" or "Remove output from punch XXXX, then ready punch" is sent and the routine awaits operator response. However, if the combine option is indicated, an IORCB for a card to be read and stacked in pocket 3 is built and executed before this message is issued.

For a punch, a blank record is constructed and PUT in order to force the last card into the stacker. If this punching is incomplete, control returns to the user with general register 15 indicating "incomplete". If the punching is complete with error, another PUT is issued to obtain error pointers for the user if this has not already been done (error recovery will be attempted if the error indicator is set off) and control returns to the caller with the Finish-Just-Issued flag on, the Finish-in-Progress flag off, and a return code indicating "incomplete with error" in general register 15. If no error occurred in the punching, processing continues, as with the printer, at the PUT for flushing the last buffer.

For subsequent entries to Finish, the path taken depends on the value of DCBRCX as set by the previous FINISH.

DCBRCX=30: Completion of input is being awaited. If the input operation is not yet complete, the routine returns to the caller with the return code indicating "incomplete". If the input operation is complete with error, the completion routine (see DCBRCX=80 below) is entered. If the input operation is complete without error processing continues at the message-defining loop. If any of the DECBs are marked complete with no errors and a unit check is indicated, the message "Remove output from reader XXXX, then ready reader" is sent and operator response is awaited. If a unit exception is indicated, the message "Remove output from reader XXXX" is sent and Finish enters its completion routine. If there are no such DECBs, the message sent will instead be "Remove input/output from reader XXXX, then ready reader."

DCBRCX=40: Finish is awaiting completion of the PUT. Processing continues as if Finish had just issued its PUT to write the last buffer page.

DCBRCX=60: Awaiting an asynchronous interruption (operator response). The interruption has still not occurred, so the return code is again set to indicate "incomplete" and the routine returns to the caller. (Not applicable to RJE.)

DCBRCX=80: End of wait for the asynchronous interruption. This condition is caused by the occurrence of the interruption or by the user's turning off the FINISH-in-Progress bit before the interruption was received. The completion routine is entered, the FINISH-Just-Issued flag is

set, the FINISH-in-Progress flag is turned off, the appropriate return code indicating "complete" is set in general register 15, and the routine returns to the caller. (Not applicable to RJE.)

DCBRCX=50: A wait for a card to be stacked in pocket 3 when the device is a punch and the combine option is indicated. If there are any outstanding IORCBs, the card-read-and-stack operation is not yet complete so the R15 return code is set to "incomplete" and the routine returns to the caller. Otherwise, if no errors were recorded on the card-read-and-stack operation, the message "Remove output from punch XXXX, then ready punch" is sent and operator response is awaited. If an error was recorded, the message sent instead will be "Feed card from reader YYYY, stack in pocket 3, remove output from punch XXXX, then ready punch." (Not applicable to RJE.)

Asynchronous Interruption Entry Point (CZCMH2) Processing: This interruption is caused by the operator changing the device state from "not ready" to "ready", the response required following the issuing of a message. (This routine is not entered during an RJE task.)

The DCBRCX field is set to indicate that the interruption has occurred. The DIR macro is used to delete the interruption routine and control is passed to the calling routine with a return code of zero in general register 15.

Synchronous Interruption Entry Point (CZCMH3) Processing: This interruption is caused by the completion of a read, feed and stack in pocket 3 operation when the combine option is indicated. (This routine is not entered during an RJE task.)

Errors occurring during the I/O activity may result in limited retry, depending upon the type of error. Any final error is recorded in the DEB.

Except when a retry is in progress, the number of outstanding IORCBs is reduced to zero and then, in all cases, a return code of zero is set in general register 15, and the routine returns to the task monitor.

MSAM Close Routine (CZCMI)

MSAM Close calls Finish to complete output if necessary, to attempt recovery from an error on a previous PUT, or to indicate end of data group to the operator. MSAM Close frees pages of virtual storage obtained by MSAM Open. (Chart DD.)

Attributes: Privileged, reentrant, read-only, public, system, nonrecursive.

Entry Point: CZCMI1 -- Entered from Common Close via type-1 linkage.

Input: Register 1 contains the address of the DCB.

Data References: CHADCB, CHADEB, CHADEC, CHATDT, CHASDA, CHAICB, CHADBP.

Modules Called:
MSAM Finish (CZCMH) -- Complete output and indicate end of data group.

DIR (CZCJD) -- Delete an active interruption routine.

FREEMAIN (CZCGA) -- Release virtual storage.

INTINQ (CZCJI) -- Interruption inquiry.

AWAIT (CEAP7) -- Await an interruption.

RJELC (via SVC 232) -- Disconnect RJE line control.

Exits:
Normal -- Return to caller.

Error -- ABEND macro used for abnormal termination.

Operation: If the DEB is invalid, MSAM Close abnormally terminates.

Finish is called to assure that I/O has been completed. If it has not, the routine goes into the wait state (using AWAIT) until the I/O is complete. FINISH will then be reissued. This process will be repeated until all DECBs have been checked for I/O completion. If Finish is awaiting an asynchronous interruption the INTINQ macro will be issued. If Finish is in the process of stacking a blank card, Close will wait until that operation is complete. The interruption will then be dispatched and FINISH will be reissued until a return code is received indicating that the operation is complete. In an RJE task, the INTINQ macro will not be issued (Finish does not field an asynchronous interruption during an RJE task) and the test for a blank card being stacked is bypassed. If an unrecoverable error occurred on a PUT, FINISH is reissued in order to attempt recovery before informing the user of the error.

The SDAT malfunction flag is set on if a permanent error is indicated. The DEB pointer in the JFCB is removed, and the DIR macro instruction is used to delete any active interruption routine. The DIR is bypassed in an RJE task (BULKIO specifies and deletes interruption routines for RJE). RJE line control will be disconnected if the device is a remote printer and the

installation operator will be notified at his console via a WTO macro.

If the DEB indicates user read-write protection class, two FREEMAIN macro instructions are issued to free the two noncontiguous groups of virtual storage pages obtained by MSAM Open. If user-read-only or user-inaccessible protection is indicated, a single FREEMAIN is issued to free the contiguous pages of virtual storage obtained by MSAM Open.

The DEB pointer in the DCB is removed, and control returns to Close Common.

TAM Close Routine (CZCYG)

TAM Close is called by Close Common if a user desires to close a TAM DCB, because of ABEND requirements for a task to be closed, or as a result of a LOGOFF command.

In continuing the close processing from Close Common, TAM Close is called to perform additonal closing functions unique to TAM terminals. This includes freeing the control blocks and buffer areas obtained during TAM Open and performing the disable/enable function at logoff time. TAM Close then returns to Close Common except when an abnormal end is required in which case it goes to ABEND or SYSER. (See Chart DE.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged.

Entry Point: CZCYG1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of a two-word parameter list:

Word 1 -- Address of DCB being closed.

Word 2 -- Address of associated JFCB.

Data References: CHADCB, CHATDT, CHASDA, CHADEB, CHAISA, CHADEC.

Modules Called:
Write (CZCYM) -- TAM write.

Check (CZCRC) -- Check.

VMA (CZCGA) -- Free virtual storage.

ABEND (CZACP) -- Abnormal task termination.

WTO (CZABQ) -- Write to operator.

SETAE (CEAAK) -- Set asynchronous entry.

ADDEV (CEAAC) -- Add device to task device list.

RMDEV (CEAAD) -- Remove device from task device list.

XTRCT (CEAH3) -- Extract TSI field.

SYSER (CEAIS) -- System error.

Exits:
Normal -- Return to calling routine.

Error -- 1. ABEND macro instruction.
2. SYSER.

Operation: TAM Close initially saves the general registers. The Recursive Call flag is tested to prevent a recursive loop between TAM Close and ABEND. If the flag is set, it indicates that this is a recursive call. This means that just prior to this entry to TAM Close (through Close Common) from ABEND, and exit from TAM Close to ABEND occurred. TAM Close therefore does not proceed but only clears the Recursive Call flag and branches to ABEND.

If the Recursive Call flag is not set, the opened DCB count in SDAT for this terminal is decremented by 1 (for this DCB) and the number of opened DCBs that are still open for this terminal is examined.

If the number is positive, the pages of virtual storage created for this DCB are freed, and the pointers that were set during TAM Open are removed. (See TAM Open Figure 4.) A return is then made to Close Common.

If the number is zero, processing continues. (This indicates there is now no opened DCB for this terminal.)

If the number is negative, the recursive call flag is set, a count of zero replaces the negative number and processing continues as if there were a zero count.

Processing continues with this terminal no longer having any opened DCB. A test is made to determine if the interruption storage area (ISA) flag is set with ABEND=2. If it is not, TAM Close continues testing the terminal to determine the type of close.

If it is set, then TAM Close immediately goes to the final steps of a close.

Note: ABEND=2 indicates that the entry to this closing came from ABEND and, after the closing, a return is made to ABEND. When this return is made, ABEND retains control of the terminal.

Testing continues at this point to determine the type of close by verifying that the terminal is defined and, if so, if it is on a 2702.

If it is defined and is not on a 2702, TAM Close proceeds with the final steps of

a close, bypassing the disable/enable function. (The terminal defined in this manner is the operator's terminal; a 1052-7 direct connection to the multiplexor channel.)

If it is supported and is also on a 2702, then the disable/enable function is required. Initially, after checking that the device has not been phased out, processing takes place to provide a SYNAD address for the disable/enable function. The SYNAD address in the user's DCB is saved in a temporary area within TAM Close and is replaced with a TAM Close SYNAD address. Should any I/O operation fail during the disable/enable function the SYNAD will either declare a minor SYSER software error or abnormally terminate the task. In all error cases the task is abnormally terminated. At this point the terminal is disabled and TAM Close must then enable the line. In order to enable the line, TAM Close reopens the DCB (by setting the DCB open flag) that was previously closed in Close Common. After performing the following functions, TAM Close closes the DCB again with the disable/enable function completed and the user's SYNAD address replaced. The following functions are required in order for TAM Close to enable the line:

- The disabled terminal must be added to the task.
- Asynchronous interruptions must be ignored when the disabled terminal is connected to the task.
- The terminal control unit must be restored to the initial SAD order.
- The line is then enabled.
- The terminal is removed from the task.

If it is not defined, then the Recursive Call flag is set and TAM Close proceeds with the final steps of a close, bypassing the disable/enable function.

In the final steps of a close, the pages of virtual storage created for this DCB are freed and the pointers that were set during TAM Open are removed (see Figure 4).

A final test is then made of the Recursive flag. If it is on, a message is issued to the operator and ABEND is invoked. If it is not on, a return is made to Close Common.

IOR Close Routine (CZCSD)

IOR Close is called by Close Common:

- Due to normal completion of a task.
- Due to ABEND requirements for a task to be closed.

In continuing the close processing from Close Common, IOR Close is called to perform additional functions for these devices. IOR Close waits until all outstanding DECBs have been completed and then frees the IOREQ work area, DEB and IORCB. IOR Close then returns to Close Common. (See Chart DF.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, nonrecursive, privileged.

Entry Point: CZCSD1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of a two-word parameter list:

Word 1 -- Address of DCB being closed.

Word 2 -- Address of associated JFCB.

Data Reference: CHADCB, CHADEC, CHADEB.

Modules Called:
VMA (CZCGA) -- Free virtual storage.

AWAIT (CEAP7) -- Await an interruption.

DIR (CZCJD) -- Delete asynchronous interruption requests pending.

Exit: Normal return to calling program.

Operation: IOR Close initially saves the general registers. The address of the DEB (DCBDEB) is obtained from the DCB. A test for any unchecked DECBs is made.

- If there are no unchecked DECBs (DEBNPC=0), processing continues by determining the area to be freed.
- If any DECB is unchecked (DEBNPC≠0), the address of the last DECB in the queue is obtained (from DEBDEL). The AWAIT SVC is moved into this DECB and the IOR Close executes (EX) the AWAIT in this DECB. IOR Close then waits until this last DECB is posted complete (DECECB), and then processing continues by determining the area to be freed.

In determining the area to be freed the size of the DEB is added to the size of the IORCB and then a FREEMAIN macro instruction is issued to free the area. The final step of IOR Close is to return to Close Common.

LABEL PROCESSORS

The following routines describe tape and direct access input and output label processors.

Tape Volume Label Routine (CZCWX)

The Tape Volume Label routine is called by Device Management to read a tape volume label, or to rewind and unload a tape. It may also be called by Device Management or the LABEL command routine to write a volume label (Chart EA).

Entry Points:
CZCWX1 -- Entered to read the volume label.

CZCWX2 -- Entered to rewind and unload a tape.

CZCWX3 -- Entered to write the volume label.

Input: Register 1 contains the address of the following parameter list:

Word 1 -- Address of the SDAT.

Word 2 -- Address of an 80-character label buffer.

Word 3 -- Address of a 1-byte field containing the density setting.

Word 4 -- Address of a 1-byte field containing ASCII/EBCDIC indicator (X'20' = ASCII; X'00' = EBCDIC).

Modules Called:
Control (CZCRBS) -- Rewind, rewind and unload, write Tape Mark, or backspace.

GETMAIN (CZCGA2) -- Get a work page.

Read/Write (CZCRAS) -- Read a block, write a label.

Exits:

Return Code	Condition
00	Normal return, or incorrect length on READ or WRITE.
04	Intercepted more than 10 times, or unit check with no file protect.
08	Unit exception.
0C	Unit check with file protect.

Operation: The text of the 'Operation' is keyed to the labels of Chart EA.

A Read is entered at CZCWX1, a Write at CZCWX3. Tape Recording Technique, density, BPI are set (CA001), a temporary DEB built (CA001A) and the tape rewound (CA002).

CZCRAS reads the label for a Read option and returns on normal completion or returns a code as described under 'Exits' (CA004C - CA008A).

CZCRAS writes a number of tape marks if the first word of the buffer is blank, or writes a label. An American National Standard format label is written where ASCII is specified. The label write will be followed by a write of tape marks. If CZCWX was called by Pause, two tapemarks are written; otherwise, five tapemarks are written (via five calls to Control) (CA004 - CA004A - CA008 - CA005E).

If CZCWX was called by PAUSE, the file is backspaced past the tape marks. If not called by PAUSE, it is rewound and unloaded (CA005F - CA005B).

At CZCWX2, a temporary DEB is created and CZCRB is called to rewind and unload the tape.

Various return codes reflect the possible READ/WRITE errors. (See 'Exits'.)

Tape Data Set Label Routine (CZCWY)

This routine reads and writes data set header and trailer labels on magnetic tape volumes. Error checking is provided after reading labels. User label routines are called if required. Provision is made for reading and writing labels in either standard IBM or American National Standard formats. (See Chart EB.)

Attributes: Reentrant, resident in virtual storage, read only, public, privileged, system, nonrecursive.

Entry Points:
CZCWY1 -- Entry point for processing header labels for input data sets.

CZCWY2 -- Entry point for processing trailer labels for input data sets.

CZCWY3 -- Entry point for processing header labels for output data sets.

CZCWY4 -- Entry point for processing trailer labels for output data sets.

Input: Register 1 contains the address of a parameter list consisting of the address of the SAM communication block, CHASCB. CHASCB includes SCBIOA, which points to an 80-byte buffer area. Fields of special interest reached through CHASCB:

DCGOFG (Open Flag - DCBO3M) which may contain:
0 Data set being opened or closed.
1 Data set open, EOV process.

DCBOFG (EOT Flag - DCBO6M) which may contain:
0 Process on basis of open flag.
1 EOT occurred while writing EOF labels - write EOV this time.

DCBOPT (Option - DCBSU2) which may contain:
0 Standard IBM labels (EBCDIC user).
1 American National Standard labels (ASCII user).

TDTLAB (Labels - TDTSUM) which may contain:
0 No user labels.
1 Process user labels.

Data References: CHASCB, CHADCB, CHADEB, CHADEC, CHAISA, CHALB1, CHALB2, CHASDA, CHATDT.

Modules Called:
BSAM Read/Write (CZCRA) -- Read or write a physical record.

Control (CZCRB) -- Provide non-data operations on the tape device.

LVPRV (CZCJL) -- Provide type-3 linkage to nonprivileged user label routines.

User Prompter (CZCTJ) -- Communicate with user.

Volume Sequence Convert (CZCWV) -- Compute the address of the volume serial field of JFCB specified by relative volume sequence number in SCBRVS.

Exits:
Normal -- Return to calling program with return code 0 in register 15.

Error -- Link to CZACP (ABEND) with register 1 pointing to message area, first byte of which contains the length of the message text.

Operation: An initialization section used in common from all four entry points performs standard linkage and save requirements and establishes addressability. A branch table containing the addresses of the four separate labeling routines is encountered, and the appropriate label processor reached based on a code saved upon entering Tape Label Processor at one of the four entry points.

Processing for each of the four labeling routines is discussed below separately. In addition, three subroutines, Build, a routine which actually builds the tape label in a buffer for output, Check, which determines processing after a label I/O operation has occurred, and SUL, which handles user labels, are described.

Input Header Label Processor (CZCWY1): The tape label is read and the read checked by a branch to the Check subroutine. If a read error has occurred, or a tape mark or the beginning of the tape is encountered, an abnormal end is made. If the label is a volume label, another read is issued. If the label is not a volume label, and is not a HDR1, EOV1, or EOF1 label, an abnormal end is made.

If the label is HDR1, EOV1, or EOF1, the block count is stored in the DCB. The DSNAM subroutine is entered to check the 17 least significant characters of the data set name against those in the JFCB. The user is notified through the PROMPT macro instruction when the data set names in the label and the JFCB do not agree and given the option to continue or to terminate by an abnormal end.

If the generation/version numbers are not correct, the user, as above, has the choice of terminating or continuing.

The next label is read and the read is checked. An abnormal end termination is made if a read error occurs. If a tape mark is read during a read backward operation, the routine backspaces the tape to position it for a data set read. If the label was an HDR2, an EOF2, or an EOV2, and if the record format, record length, and/or block size in the CHADCB are zero, the routine obtains the information from the label, converts it to binary, and stores it in the DCB. If the user has indicated his data set is in ASCII, the HDR2 label will have a buffer offset field. Unless this field contains zero, it will also be converted to binary and stored in an appropriate field in the DCB.

If standard user labels are not specified, or if they are specified but the user input header label exit is not active, the tape is positioned to read the data set, and a normal return is made.

With user labels specified, a branch is made to the SUL subroutine (explained below). On return to the Input Header routine, the tape is positioned to read the data set, and a normal return is made.

Input Trailer Label Processor (CZCWY2): This routine processes standard IBM (EBCDIC users) and American National Standard (ASCII users) trailer labels for an input data set being read forward, and header labels for an input data set being read backwards. User labels are also processed.

The routine first reads and checks the label. If an error or tape mark is encountered during the read operation, or if the label read was invalid, the abnormal end code is set in the CHASCB, and an abnormal end is effected.

If a volume label was read, the read is reissued.

If the block count of the CHADCB does not equal that in the label, the user is informed, and a reply is expected. The user may elect to terminate the task.

If standard user labels are specified, and the exit is active, another READ is issued and checked. Again, if a read error was encountered, the abnormal end code is set and an abnormal end is effected. The READ is reissued if an HDR, EOF, or EOV label is encountered. Otherwise, the user is given the facility to process the label.

User labels are read and checked by the SUL subroutine.

Output Header Label Processor (CZCWY3): This routine checks the currently mounted output tape to see if the expiration date of any data set currently on the tape (and about to be overlaid) has been reached and oversees creation and writing of new output header labels in either standard IBM or American National Standard (ASCII users) format. It sets up the mechanism for writing up to eight standard user labels, or an unlimited number of user labels when ASCII is specified.

On entry to this routine, the tape label is read and checked. If a volume label is encountered, the next label is read; this label should be either a HDR1 label (from an old data set which we are overlaying) or a tape mark (if the tape is clean). Where neither is encountered, abnormal termination occurs.

If HDR1 was present, the expiration date of that data set is checked and if it has not yet been reached, the user is prompted. He may choose to continue or request abnormal termination.

The logical file (data set) sequence number is taken from the JFCB, the tape is repositioned to write the header labels, and the Build subroutine called to write both HDR1 and HDR2 labels.

If user labels have been requested, a branch is made upon return from Build to the SUL subroutine.

On return from SUL, the Control subroutine is called to write a tape mark following the header label group and position the tape to write the data set. Normal return is made to the caller.

Output Trailer Label Processor (CZCWY4): If CLOSE processing is being done, the characters EOF1 are placed in the first four bytes of the 80 character label buffer. Otherwise, this routine must have been called for an end-of-tape condition, and an EOV1 is placed there. The remainder of the label is created and written by a branch to the Build subroutine. Build writes the two trailer labels in either standard IBM or American National Standard label format.

The SUL subroutine is called if user labels have been requested. Control is called to write a tape mark and position the tape. Normal return is then made to the caller.

Build Subroutine: This routine builds 80-byte output header and trailer labels for tape volume data sets and writes them by calling BSAM Read/Write.

The routine expects the first four bytes of the 80-byte label buffer to already contain HDR1 (for header labels), or EOF1 or EOV1 (for trailer labels). Build fills in the remaining bytes as shown in Table 21.

Initially, after moving the 17 least significant characters of the data set name to the label buffer, the Volume Sequence Convert routine is called to compute the address of the volume serial field of the JFCB. The data set serial number is obtained from this field if the volume is not mounted, or from the SDAT if it is.

Then, in order, the relative volume sequence number, the data set sequence number, generation and version numbers, creation and expiration dates, block count, and system code are moved into the label buffer. The data set security number field is made blank for American National Standard (ASCII users) label format or a character zero for standard IBM format.

Build then branches to the Write subroutine in Tape Data Set Label which calls BSAM Read/Write to write the first label on the tape. The Check subroutine determines tape write errors or end of tape and passes that information back to Build. Build now sets the label number to two and makes up the second label, illustrated in Table 22. The standard IBM and the American National Standard second label formats are similar

Table 21. Label 1 Fill Table

Label Field	Size In Bytes	Filled From	Remarks
Data Set Identifier	17	CHATDT (JFCB)	17 least significant characters of 44 character Data Set Name in the Job File Control Block
Data Set Serial Number	6	CHASDA or JFCB	Filled from Volume ID field of CHASDA if volume is mounted, otherwise from Volume ID field of first Volume Serial field in the Job File Control Block
Volume Sequence Number	4	CHASCB	
Data Set Sequence Number	4	CHATDT (JFCB)	
Generation Number	4	CHATDT (JFCB)	
Version of Generation	2	CHATDT (JFCB)	
Creation Date	6	CHATDT (JFCB)	
Expiration Date	6	CHATDT (JFCB)	
Data Set Security Number (Accessibility)	1	--	Not Implemented (blank if ANS; zero if standard IBM)
Block Count	6	CHADCB	
System Code	13	--	
Reserved	7	--	Filled with blanks
TOTAL	76 Bytes		

Table 22. Label 2 Fill Table

Label Field	Size In Bytes	Filled From	Remarks
Record Format	1	CHADCB	
Block Length	5	CHADCB	
Record Length	5	CHADCB	
Density	1	CHADCB	
Data Set Position	1	--	Contains 0 if first volume mounted, otherwise contains 1
Job/Step Identification	17	--	OS only
Tape Recording Technique	2	CHADCB	
Print Control Character	1	CHADCB	
Reserved	13	--	Contains Blanks
Buffer Offset	2	CHADCB	ANS labels (ASCII) only; otherwise reserved
Reserved	28	--	Contains blanks
TOTAL	76 Bytes		

except for the buffer offset field which must be filled in for the American National Standard format.

The routine effects an abnormal end if there is a tape write error in writing either label. If, on attempting to write a header label, an end-of-tape indication is detected, an EOV trailer label is written instead and the header label is written on the next volume.

A more detailed discussion, including tables, of both standard IBM and American National Standard label formats, is contained in Appendix A of Data Management Facilities, GC28-2056.

SUL Subroutine: This routine reads and writes user labels. Standard IBM format users may have a maximum of eight user labels; ASCII users, who must conform to the American National Standard label format, are allowed in TSS/360 to have a number of user labels limited only by the physical extent of the tape volume.

Initially, the routine calling SUL will have placed UHL1 (user header label) or UTL1 (user trailer label) in the label buffer. SUL will then determine from a code passed in register 1 whether to read or write. If a user label is to be read, the Read subroutine will be branched to for reading and checking a user label. An abnormal termination will occur if the label read is neither a header or trailer label. The user's label processor will then be called.

Where a label is to be written, the user's label processor will be called first to build the rest of the label in the label buffer, and then SUL will cause the label to be written and checked.

If the user is nonprivileged, his user routine is not called directly; he gains control through the Leave Privilege routine.

When the user labels have all been read or written (a maximum of eight for EBCDIC users, no maximum for ASCII users), return is made to the routine which called SUL.

Check Subroutine: This routine is entered following a return from BSAM Read/Write to check the I/O operation results and determine further processing.

Check inspects various fields and flags in the data event control block and sets the return code as follows:

OPERATION CHECKED		Rtn.
READ	WRITE	Code
Normal Complete	Normal Complete	'00'
Unit Exception	--	'04'
Error	Error	'08'
Intercepted	Intercepted	'0C'
Unit Check (Beginning of Tape)	Unit Exception	'10'

If the operation is intercepted, the routine sets a counter so that a maximum of 50 interceptions on this operation is allowed before the return code is set to the error code.

If the operation is not complete, the AWAIT macro instruction is used to allow the operation to complete.

DA Input Label Routine (CZCXN)

The Direct Access Input Label processor reads standard direct access user header or trailer labels, and provides linkage to a user label processing routine as specified in the DCB exit list. (See Chart EG.)

Attributes: Reentrant, nonrecursive, resident in virtual storage, privileged.

Entry Point: CZCXN1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the SAM communication block.

Data References: CHASCB, CHADEB, CHADCB, CHAISA.

Modules Called:
Message Writer (CZCWM) -- Message processing and ABEND processing.

Obtain/Retain (CZCFO) -- Obtain DA user label.

LVPRV (CZCJL) -- Leave privileged state.

Exit:
Normal -- Return to caller.

Error -- Call Message Writer to do ABEND processing.

Operation: Standard user labels are read via Obtain, and, if the user is non-privileged, processing is provided for via the Leave Privilege routine (LVPRV). Figure 8 shows Obtain keys and Label I/O

UHL (n) or UTL (n)	UHL (n) or UTL (n)		CCHHR000
Obtain key	User label key	Miscellaneous User Data	Direct Access Address
4	4	76	8

Figure 8. Obtain Keys and Label I/O Areas

area. When a file mark is read, or the user returns a zero code from LVPRV, the routine places a zero code in general register 15 and returns control to the invoking program.

If the exit type in SCBEXT is not for input user header or trailer labels, the routine sets a unique ABEND code in SCBABN, and links to the Message Writer for ABEND termination.

If there was a hardware error while attempting to read a label, the message "XXXX LABEL UNREADABLE, ENTER N TO READ NEXT LABEL, B TO BYPASS LABELS OR E TO END THE TASK" is sent to the user via the Message Writer routine. If the user replies "N", and there have not been eight attempts to read labels, reading of the next label is attempted.

If eight attempts have been made, or if the user replies "B", control is returned to the caller with a zero return code in general register 15. If the user replies "E", the routine sets a unique ABEND code in SCBABN and calls the Message Writer routine to perform ABEND termination.

DA Output Label Routine (CZCXU)

The Direct Access Output Label processor provides the user with the facility for building and writing standard user labels on a direct access device. (See Chart EH.)

Attributes: Reentrant, nonrecursive, resident in virtual storage, privileged.

Entry Point: CZCXU1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the SAM communication block.

Data References: CHASCB, CHADCB, CHADEB.

Modules Called:
LVPRV (CZCJL) -- Leave privileged state.

Obtain/Retain (CZCFO) -- Retain DA user label.

Exits:
Normal -- Return to caller with '00' in general register 15.

Error -- Register 15 contains '04'.

Operation: If there is a label track in the first extent of the data extent block, and an output label type is specified in the CHASCB, and the exit type is active, the user is given the facility to build labels via the Leave Privilege routine, and to write labels via Retain, until the user passes a zero return code in general register 15, or until eight labels are written. At that time, a file mark is written via Retain. (See Figure 9 for the RETAIN keys and the LABEL I/O area.)

If the exit type specified in the CHASCB is not for output header or trailer labels, a unique ABEND code is set in the CHASCB, and control is returned to the caller with a return code of '04' in general register 15.

If Retain returns with a code other than zero in general register 15, a unique ABEND code is set in the CHASCB, and control is returned to the caller with a return code of '04' in general register 15.

EOV PROCESSORS

End of volume processing consists of tape and direct access input and output EOV routines, mainline and 'forced' EOV routines, and concatenation and check processors. The EOV routines are entered as the result of one of two conditions: end of data set or end of volume.

UHL (n) or UTL (n)	UHL (n + 1) or UTL (n + 1)		CCHHR000
Retain key 4	User label key 4	Miscellaneous User Data 76	Direct Access Address 8

Figure 9. Retain Keys and Label I/O Areas

Force End of Volume Routine (CZCLD)

The Force End of Volume routine terminates the processing of the current volume of a data set, and prepares for the processing of the next volume. It accomplishes this by initiating the End of Volume (EOV) routines. (See Chart FA.)

Attributes: Reentrant, resident in virtual storage, closed, privileged, read-only, public.

Entry Point: CZCLDF -- Entered by type IM or type IIM linkage.

Input: Register 1 contains the address of the DCB.

Data References: CHADCB, CHADEB.

Modules Called:
SAM Mainline EOV (CZCXE) -- SAM Mainline end of volume processor.

QSAM FEOV (CZCSA) -- QSAM forced end of volume processor.

Exits:
Normal -- Return to calling routine.

Error -- ABEND macro instruction.

Operation: Force End of Volume abnormally terminates the task via the ABEND macro instruction for any of the following reasons:

- DCB identifier is not valid.
- Data set is not physical sequential.
- Data set is not on magnetic tape or DA.
- Not all BSAM DECBs have been checked.

Control is returned to the calling routine when the input DCB is not open.

Should all of the above tests be passed and if QSAM, QSAM FEOV is entered. If not QSAM, or on return from QSAM FEOV, the FEOV flag is set on and the Mainline EOV routine is entered. When Mainline EOV returns to Force End of Volume, control is then returned to the calling routine.

Note the function of this routine is mainly performed by Mainline EOV.

Mainline EOV Routine (CZCXE)

Mainline EOV performs as a control program to End of Volume (EOV) processing. It oversees the modifying of the DEB, label processing, volume switching, determining end of data set, concatenation processing,

and passing control to user label exit routines. (See Chart FB.)

Attributes: Reentrant, resides in virtual storage, closed, read-only, privileged.

Entry Point: CZCXE1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the DCB causing EOV involvement. The PSECT contains CHASCB and three temporary control blocks for use in the I/O operations of label processing.

Data References: CHADCB, CHADEB, CHASCB, CHATDT, CHADEC, CHAISA.

Modules Called:
VMA (CZCGA) -- Get virtual storage, free virtual storage.

Build Common DEB (CZCWB) -- Restore the common portion of the DEB.

Volume Sequence Convert (CZCWV) -- Volume address convert.

DA Input EOV (CZCXD) -- Direct access output end of volume processing.

DA Output EOV (CZCXI) -- Direct access input end of volume processing.

Tape Output EOV (CZCXO) -- Tape output end of volume processing.

Tape Input EOV (CZCXT) -- Tape input end of volume processing.

User Prompter (CZCTJ) -- Communicate with user.

AWAIT (CEAP7) -- Await an interruption.

Exits:
Normal -- Return to calling program.

Error --

- Call User Prompter to inform user.
- Via ABEND.

Operation: The SAM control block (CHASCB) is initialized by zeroing the variable fields, storing pointers to the input DCB, JFCB, and DEB, and storing the volume address in the volume serial field of the SCB. The pointers to the JFCB and DEB are known because the input DCB contains a pointer to the DEB, and the DEB points to the JFCB.

Mainline EOV assures the DEB pointed to by the DCB has a valid DEB-id, and that the DEB points to the DCB passed. If not, the routine abnormally terminates.

The routine initializes the temporary DCB and DEB for use by SAM routines called by SAM EOV. End of data set is indicated for non-concatenated unit-record data sets.

If the device is tape, the temporary control blocks are adjusted so tape labels may be read or written. For DA devices, the temporary DCB is set so user labels may be read or written. In either case, tape or DA, if user labels are specified, GETMAIN is called to get unprotected virtual storage for a label buffer.

Depending on whether the device is tape or DA, and whether the last I/O operation was input or output, Mainline EOV branches to Tape Output EOV, Tape Input EOV, DA Output EOV, or DA Input EOV routines.

When the device-oriented EOV routine returns control to Mainline EOV, any storage which had been obtained for label buffers and format-3 DSCBs is released via FREEMAIN.

If no abnormal conditions occurred during EOV processing, control is returned to the calling routine.

Tape Input EOV Routine (CZCXT)

The Tape Input EOV routine executes the end-of-volume procedures for the mounted input magnetic tape volume. That is, it oversees label processing and final volume positioning. If the mounted volume is only one of a multivolume data set, Tape Input EOV causes the next volume to be mounted, oversees positioning and label processing of the new volume, and updates the DEB to reflect the presence of the new volume. If the data set is a member of a concatenation of data sets, the Concatenation routine is used. (See Chart FC.)

Attributes: Reentrant, resides in virtual storage, closed, read-only, privileged.

Entry Point: CZCXT1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block.

Data References: CHADCB, CHADEB, CHASCB, CHATDT.

Modules Called:
Control (CZCRB) -- Tape positioning.

Bump (CZCAB) -- Request and verify mount of new volume.

Build Common DEB (CZCWB) -- Modify the common portion of the DEB.

Tape Data Set Label (CZCWY) -- Tape label processor.

BSAM Read/Write (CZCRA) -- Read ahead for tape mark.

User Prompter (CZATJ) -- Send message to user.

Tape Positioning (CZCWP) -- Tape positioning.

Volume Sequence Convert (CZCWV) -- Volume address conversion.

Concatenation (CZCXX) -- Concatenation.

Exits:
Normal -- Return to calling program.

Error -- Call Message Writer to do ABEND processing.

Operation: For forced end-of-volume conditions, Tape Input EOV proceeds as if the EOV condition was encountered for unlabeled tape.

The tape is properly positioned via the Control routine.

When trailer labels are present, Tape Data Set Label is called at entry point CZCWY2 to process user trailer labels and standard trailer labels.

Unless a read backward operation was performed, a READ is issued to check for a second tape mark, indicating end of tape. Otherwise, end of data set processing is required. The tape is repositioned via CONTROL.

If this is the last volume, and there is no concatenation, the End of Data Set flag is set in the DCB and control is returned to the calling routine.

The Concatenation routine is invoked if concatenation is indicated. For a non-end-of-data-set condition, the Bump routine is used to mount the next volume. The DEB is updated via Build Common DEB to reflect the new volume and the Tape Positioning routine is used to position the tape for processing. If the newly mounted tape has lables, the Tape Data Set Label is called at entry point CZCWY1 to process standard labels and user header labels. Tape Input EOV then returns control to the calling routine.

The routine abnormally terminates when Bump indicates that the requested volume has not been mounted, or when the relative volume sequence of the volume to be mounted is less than the volume sequence number of the current volume.

Tape Output EOV Routine (CZCXO)

When a data set is being closed, Tape Output EOV oversees the end-of-volume tape processing, including the writing of trailer labels. For multivolume output data sets, this routine oversees the mounting of the new volume and updating of the DEB. (See Chart FD.)

Attributes: Reentrant, resides in virtual storage, closed, read-only, privileged.

Entry Point: CZCXO1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block which was generated in the PSECT of Mainline EOV.

Data References: CHADCB, CHASCB, CHATDT.

Modules Called:
Control (CZCRB) -- Tape positioning.

Bump (CZCAB) -- Request and verify mount of new volume.

Build Common DEB (CZCWB) -- Modify the common portion of the DEB.

Volume Sequence Convert (CZCWV) -- Volume address conversion.

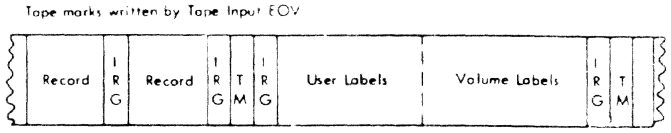
Tape Data Set Label (CZCWY) -- Tape label processor.

User Prompter (CZCTJ) -- Communicate with user.

Exits:
Normal -- Return to calling program.

Error -- Via ABEND macro instruction.

Operation: Tape Output EOV turns off the FEOV flag in the DEB and, via the Control routine, writes a tape mark on the tape. If labels are specified, the Tape Data Set Label is called at CZCWY4 to write user and standard volume trailer labels. Another tape mark is written and the volume positioned immediately before the tape mark.



If the data set is being closed, normal return is made to the user.

Otherwise, the address of the next volume is obtained by Volume Sequence Con-

vert, the volume just ended is rewound by Control, and the volume is mounted by the Bump routine. The DEB is updated via Build Common DEB to indicate the presence of the new volume and if labels are specified, the user and standard header labels are processed by the Tape Data Set Label.

Where an end-of-volume indication is received while the EOF trailer label is being written, the trailer label is rewritten to contain EOV instead. The EOF trailer label is then written by the Tape Data Set Label at the beginning of the newly mounted volume following its volume and header labels.

This routine abnormally terminates when Bump returns an error code indicating that the requested volume was not mounted, or when there is an incorrect volume sequence number.

DA Input EOV Routine (CZCXI)

When a data set is being closed, Direct Access Input EOV oversees the end-of-volume direct access device processing, including the reading of user trailer labels. For multivolume input data sets, or data set members of a concatenation, this routine oversees the end-of-volume processing for the current volume, the mounting of the new, or next volume, and the updating of the DEB. (See Chart FE.)

Entry Point: CZCXI1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the SAM communication block.

Data References: CHASCB, CHATDT, CHADSC, CHADEB, CHADCB.

Modules Called:
Bump (CZCAB) -- Request and verify mount of new volume.

Obtain/Retain (CZCFO) -- Obtain DA user label and retain DA user label.

Build DA DEB (CZCWL) -- Build direct access DEB.

User Prompter (CZATJ) -- Communicate with user.

SETDSCB (CZCXS) -- Update DSCB and write file mark.

Volume Sequence Convert (CZCWV) -- Volume address conversion.

DA Input Label (CZCXN) -- Direct access input user label processor.

Concatenation (CZCXX) -- Concatenation of data sets.

Exits:
Normal -- Return to caller.

Error -- Call User Prompter or ABEND.

Operation: If the integrity bit is on, or if a write was previously executed on the volume on which the EOV condition was encountered, a call to SETDSCB is provided to rewrite the format-1 DSCB with the integrity bit off and also to write a file mark if this is indicated.

Input user labels are processed by a call to the DA Input Label routine.

If all volumes have been processed and concatenation is not specified, end of data set is indicated.

Otherwise, the Concatenation routine is called if concatenation is specified. If all volumes have not been processed, Bump is used to switch volumes. When the new volume is mounted, Obtain and Retain are called to read the DSCB and write the new DSCB respectively. The user is warned and given the option to continue if the integrity bit for the newly mounted volume is on. The new DEB is built by Build DA DEB.

DA Output EOV Routine (CZCXD)

When a data set is being closed, Direct Access Output EOV performs end-of-volume processing, including the writing of user labels and release of available unused storage. For multivolume output data sets, this routine oversees the acquisition of new extents and the updating of the DEB. (See Chart FF.)

Attributes: Reentrant, resides in virtual storage, closed, read-only, privileged.

Entry Point: CZCXD1 -- Entered by type-1 linkage.

Input: Register 1 contains the address of the SAM communication block.

Data References: CHADEB, CHADCB, CHATDT, CHASCB, CHADSC.

Modules Called:
Bump (CZCAB) -- Request and verify mount of new volume.

Extend (CZCEX) -- Allocate additional DA space for a data set.

Obtain/Retain (CZCFO) -- Obtain direct access user label and retain direct access user label.

Build DA DEB (CZCWL) -- Build or modify a direct access DEB.

Message Writer (CZCWM) -- Message processing and ABEND processing.

Read Format-3 DSCBs (CZCWR) -- Read and chain format-3 DSCBs.

Volume Sequence Convert (CZCWV) -- Volume address convert.

Set DSCB (CZCXS) -- Update DSCB and write file mark.

DA Output Label (CZCXU) -- Write user labels for DA output data sets.

Exits:
Normal -- Return to calling program.

Error -- Call Message Writer to do ABEND processing.

Operation: The integrity bit is set on. If the data set is being closed, a request is issued for space or a volume switch.

DA Output EOV uses Set DSCB and DA Output Label to write a file mark on the volume, reset the integrity bit, and provide for user output label processing.

Old data sets which exist on more than one volume and for which there is a succeeding volume, have space allocated to them on the succeeding volume.

On a forced EOV, DA Output EOV forces a volume switch.

For new data sets, DA Output EOV attempts to get space on the currently mounted volume. If it cannot:

- For private data sets - If the data set is OLD or new, and this is the last volume of the data set, a demount/mount is requested, and allocation on the newly-mounted volume is attempted.

When space is denied on the current volume, Set DSCB is used to write a file mark on the volume and to reset the format-1 DSCB integrity bit when volume switching is required. DA Output EOV uses the DA Output Label to allow user label processing.

Build DA DEB is used to extend the DEB if space is allocated on the current volume; otherwise, when space is allocated on a new volume, Build DA DEB builds a new DEB.

Concatenation Routine (CZCXX)

The Concatenation routine is called to make the next data set of a concatenated group ready for processing. (See Chart FG.)

Attributes: Reentrant, resident in virtual storage, closed, read-only, privileged.

Entry Point: CZCXX1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the SAM communication block.

Data References: CHASCB, CHADCB, CHATDT, CHADEB.

Modules Called:
Bump (CZCAB) -- Request and verify mount of new volume.

Open Common (CZCLA) -- Common open.

Close Common (CZCLB) -- Common close.

ABEND (CZACP) -- If bad return code from Bump.

Exits:
Normal -- Return to the calling routine.

Error -- Call ABEND.

Operation: The Bump routine is called to mount the first volume of the next data set.

Close Common is called to close the current data set and Open Common is called to open the next data set. A return code of four is provided in the calling program.

The routine abnormally terminates when Bump returns an error code.

BSAM USER ROUTINES

The following routines provide user control of some aspects of direct access and tape devices.

Note Routine (CZCRN)

The Note routine provides the user with identification of the last record read or written. A relative track and record number is returned, if the device is direct access, and a block count is returned for tape. All DECBs must be checked by the Check routine before Note is used. (See Chart GA.)

Attributes: Reentrant, resident in virtual storage, closed routine, nonrecursive.

Entry Point: CZCRNA -- Entered via type-1M linkage.

Input: Register 1 contains the address of the DCB.

Data References: CHADCB, CHADEB.

Modules Called:
FULREL (CZCRS) -- Convert full DA address to relative address.

User Prompter (CZATJ) -- Communicate with user.

Exits:
Normal -- Return to the calling routine.

Error -- ABEND termination.

Operation: Any data set which is not DA will be treated as if it were on magnetic tape. No error occurs when the Note routine is entered for other than DA or magnetic tape, but the address Note returns will not be meaningful for other devices.

The routine ABENDs if the DCB is invalid. If all I/O has not been checked, the user is prompted to see if he wishes to continue.

To obtain the relative address of the last record read or written on DA, the DA address of the last record is passed to the FULREL routine which converts the full DA address to a relative address. The relative address is placed in register 1, and if the last I/O operation was a write, the number of bytes remaining on the current track is placed in register 0.

For magnetic tape the Note address returned to the calling routine is a relative block count. The block count is zero when the data set is opened to write. The first block is block 0, the second block is block 1, the third block is block 2, etc. The relative block count is maintained (for the current record) in the DCBBLK.

Table 23 indicates the block count which is set into register 1 for the calling program. The flag DEBRSB indicates if the last block processed was read backwards (ON) or not (OFF).

Note: For magnetic tape volumes, if the Note routine is entered after a Point macro instruction was executed and there is no READ or WRITE macro instruction between them, it returns the current relative block count plus one (if last I/O opeaation was a read backward), or minus one (if the last I/O operation was forward). The reason for this is that there is no change in the DCBBLK field since the last READ or WRITE, yet when Note is again entered it will perform the calculation (adding or subtracting one) on the relative block count anyway.

Table 23. Decisions for Setting Block Count

Last block read backward	Y	N	N
DCBBCK=0		Y	N
Add one to block count	X		
Subtract one from block count		X	
Return with block count in register 1	X	X	X

Point Routine (CZCRM)

The Point routine performs certain operations which cause the next read or write operation to be performed at a specified block in the current volume. The relative address of the block in question is passed to this routine as an input parameter. The Point routine may be thought of as a logical repositioning of the data set. (It is a real repositioning in the case of magnetic tape.) Point may only be used with DA and magnetic tape volumes. (See Chart GB.)

Attributes: Reentrant, resident in virtua storage, closed routine, privileged.

Entry Point: CZCRMA -- Entered via type-1 linkage.

Input: The following parameters are passed:

Register 0 -- Relative DA address, or relative magnetic tape block address.

Register 1 -- Address of DCB.

The relative block address should have been obtained by use of the Note routine.

Data References: CHADEB, CHADCB.

Modules Called:
RELFUL (CZCRR) - Convert a relative DA address to a full DA address.

Control (CZCRB) -- Tape positioning.

Exits:
Normal -- Return to calling program.

Error -- ABEND termination.

Operation: Point does not apply to devices other than DA or magnetic tape. Therefore, for other devices, normal return to the calling routine is made immediately.

Note always returns the relative address of the last block read or written. If the user wishes to point to that block, the Z byte should remain zero.

When Point is entered for a DA device, the above mentioned flag (DEBIPT) is set if necessary, and the relative DA address is passed to the RELFUL routine. The RELFUL routine converts the relative address to a full DA address which is stored into the DEBNIO field. The DEBBP flag is set on to indicate to Read/Write that a Point operation has occurred.

When Point is entered for magnetic tape, if the tape volume is not currently at the requested postion, the Control routine is used to forward space or backward space the tape to the requested position. The difference between the requested block count and the current block count is the number of blocks to be skipped.

If errors occur during a point, DEBNF1 is turned on so that subsequent reads or writes will be intercepted and SYNAD will be given control when those reads or writes are checked.

Backspace Routine (CZCRG)

The Backspace routine backspaces a physical record on the current magnetic tape or direct access volume. (See Chart GC.)

Attributes: Reentrant, resident in virtual storage, closed routine, nonrecursive, privileged.

Entry Point: CZCRGA -- Entered by type-1M or type-2 linkage.

Input: Register 1 contains the address of the DCB.

Data References: CHADCB, CHADEB, CHASDA.

Modules Called:
Find Records per Track (CZCRQ) -- Find records per track.

Control (CZCRB) -- Tape positioning.

Exits:
Normal -- Return to the calling routine.

Error -- For unsuccessful completion, general register 15 contains a return code of '04'.

ABEND is called under the following conditions:

- Invalid DCB
- Unchecked Read or Write outstanding
- Illegal device or overflow records.

Operation: For magnetic tape a backspace is much easier to perform since only the physical position of the tape need be established. On the other hand, the records of a sequential data set on DA are not necessarily packed on the volume, and are not necessarily on the same track. They must be on the same volume for backspace.

For a backspace on magnetic tape, the Backspace routine links to Control via a type-1 linkage, with a backspace record request. If a backspace record goes over a tape mark, a forward space record command is issued to Control, and an error return is given to the user. If an unrecoverable error occurs, a return code of '04' is placed in general register 15 upon return to the user.

The backspace operation for a direct access volume may be very easy or quite involved, depending on where (logically) the data set is positioned. The DEB contains the DA address of the next record to be processed (DEBNIO). The Read/Write routine will operate upon that address when it is entered. Consequently, the DA backspace is accomplished if the DA address of the previous record can be stored into DEBNIO. A backspace is not performed on a data set which has track overflow specified. Neither is a backspace performed if the track containing the user labels would be entered.

If I/O operations have not been performed on the data set, no backspace is possible and control is returned to the calling routine with a non-zero return code.

Should DEBBP be zero, no prior backspace has occurred since the last READ or WRITE. And if DEBLIOR contains some positive number, there is no possibility of backspacing into labels or out of extents. Therefore, the backspace is accomplished by moving the last I/O address to the next I/O address.

If a previous backspace operation resulted in an error, control is returned to the calling routine (return code = 0) without any attempt to backspace. The previous error will be detected by the Check routine when the next I/O operation is checked.

DEBBP is set to 1 when the first backspace is made. If a previous backspace has occurred and current position is not at the first record in current track, the backspace is accomplished by subtracting from DEBNIOR which logically positions to the previous record in the current track.

Note 1: The Find Records per Track routine is used to count the number of records in a track. To use it one stores the MBCCHH of

interest in DCBRD and calls Find Records per Track. When Find Records per Track returns, the number of records in the track is in DCBRDR. If Find Records per Track is called by Backspace and it does not work properly, Backspace sets DEBNF1 on so the DECB of the next read or write will be intercepted by the Read/Write routine and Check will transfer to SYNAD.

Now, if Backspace is entered and DEBNIOR=1, the last record of the previous track must be found to accomplish the backspace. The important thing to determine is whether the previous track is just a track containing records or if it has the user labels. DEBEHT set to one indicates that there are labels in the first track of the first extent. DEBNIO pointing to the first track of the first extent means a backspace would leave the extents. DEBNIO pointing to the second track of the first extent means a backspace will get into the label track if there are labels. If the first extent is only one track in length, it will contain labels if there are labels. When DEBNIO is in the first extent and neither the first nor the second track is pointed to by DEBNIO, then it is safe to backspace to the last record on the previous track.

Note 2: The Read/Write routine will perform I/O from the DEBNIO address if DEBBP is on, and will then turn DEBBP off.

DEBBP is turned on by the Backspace routine and DEBNH is also turned on if Backspace encounters error conditions. It is assumed that Read/Write will intercept an I/O request to the data set when both the above bits are on, and that Check will transfer to SYNAD when checking that DECB.

Control Routine (CZCRB)

The Control routine performs magnetic tape positioning, card reader stacker selection and/or printer channel skipping. Control builds an IORCB containing appropriate CCWs, and executes it via the IOCAL SVC which invokes the I/O Supervisor. (See Chart GD.)

Attributes: Reentrant, resident in virtual storage, closed routine, privileged, nonrecursive.

Entry Point: CZCRBS -- Entered via type-1M or type-2 linkage.

Input: The following parameters are passed:

Register 0 -- Two-character operation code and count modifier.

Register 1 -- Address of DCB identifying I/O device.

Data References: CHADCB, CHADEB, CHAIOR, CHADEC, CHASDA.

Modules Called:
AWAIT (CEAP7) -- Await an interruption.

IOCAL (CEAA0) -- I/O call.

Exits:
Normal -- Return to calling routine.

Error --

- Abnormal termination via ABEND macro instruction.
- Exit to SYNAD routine.

Operation: Initially, the IORCB is cleared. Then it is completely filled in with the appropriate CCW as well as the address of the SAM Posting routine.

When the IORCB is complete, the IOCAL is executed. If the requested operation is REW or RUN, control is then returned to the calling routine. However, for all other requested operations the DECB is tested to determine if the operation is complete. If not complete, the AWAIT macro instruction is executed to wait for completion of this event. When the initiated operation is completed, the DECB is tested for error indicators.

Upon successful completion, control is returned to the calling routine. For unsuccessful completion, the user's SYNAD routine is given control. If there is no SYNAD routine and errors exist, the task is abnormally terminated.

ASCII Translation and Conversion Routine (CZCWA)

Users may read or write physical sequential data sets encoded in ASCII with ANS (American National Standard) label and record formats provided the storage medium is magnetic tape. Since TSS/360 processes internally in EBCDIC and standard IBM label and record formats, this routine is required to provide an interface for ASCII users. On input, this routine translates ASCII to EBCDIC and converts ANS formats to the standard IBM formats. On output, it translates EBCDIC to ASCII and converts standard IBM formats to ANS formats. (See Chart GE.)

Attributes: Reentrant, resident in virtual storage, read-only, privileged, nonrecursive.

Entry Point: CZCWA1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of a three-word parameter list:

Word 1 - Byte 1 - X'C1' for output
X'C5' for input

Byte 2 - X'00' (unused)

Bytes 3 and 4 - Length of record or block

Word 2 - Address of buffer area

Word 3 - Address of DCB

Modules Called: None.

Exits:
Normal --

- Return to caller via BR 14 with return code 0 in register 15.
- On input only, register 1 on return will contain the number of bytes shifted to overlay any block prefix.

Error -- Return to caller via BR 14 with one of the following return codes in register 15:

X'04' - First byte of parameter list not X'C1' or X'C5'

X'08' - Buffer offset greater than 99

Tables and Work Areas: CHADCB, CZCWZA (ASCII-to-EBCDIC translation table), CZCWZE (EBCDIC-to-ASCII translation table).

Operation: On input, after reading a record, where the user has defined his data set as ASCII, SAM Posting and Error Retry calls this routine to translate the record to EBCDIC. On output, before writing a record, BSAM Read/Write calls this routine to translate from EBCDIC to ASCII.

ASCII Translation and Conversion provides:

1. A character-for-character translation interface between EBCDIC and ASCII.
2. Conversion of values in block and record descriptor fields to unpacked decimal (output) or binary (input) where records are format-D (variable-length).
3. On input, evaluation of block or record format and resultant shifting of records to overlay any block prefix.
4. On output, a block prefix of 4 bytes if specified by the user.

American National Standard record formats provide for an optional block prefix which may precede the first or only logical record in each block. This prefix may contain user data and, for format-D (variable-length) records, the block length in the block descriptor field. The ASCII user, entering data sets from tape, may tell the system (in either his DDEF command or his label) to expect a block prefix; he may specify up to 99 bytes. Any data in the block prefix, other than the block length, will not be available to him, however. ASCII Translation and Conversion saves the value (the block length) in the block descriptor field (the last four bytes of the block prefix) and then shifts the first or only record left, overlaying the block prefix. The end of the record is zeroed out. On output, a user may specify only 0 or 4 bytes of block prefix, and then only if format-D records are specified. When a 4-byte block prefix is specified, it will contain a block descriptor.

The length of the block prefix is specified by the buffer offset. The differences between American National Standard and standard IBM label formats are slight; one difference is the existence of a buffer offset field in the second header label (American National Standard). If the user specifies or defaults the buffer offset (BFOFF) parameter in the DDEF command, the buffer offset field in the label determines for ASCII Translation and Conversion the number of bytes of input block prefix to handle.

The translation and conversion interfaces provided by this routine are illustrated in Figures 10 and 11.

Additional information on label and record formats, both standard IBM and American National Standard, for magnetic tape volumes is contained in Appendix A of Data Management Facilities, GC28-2056.

BUFFERING SERVICES

The following four routines are provided to allocate a pool of buffers and permit easy access and release of the buffers within the pool.

GETPOOL Routine (CZCMB)

The GETPOOL (get a buffer pool) routine fills the buffer length and the number of buffers field in the DCB. (See Chart HA.)

Attributes: Reentrant, resident in virtual storage, closed routine, read-only, public.

Entry Point: CZCMBG equated to SYSMBG -- Entered via type-1 linkage.

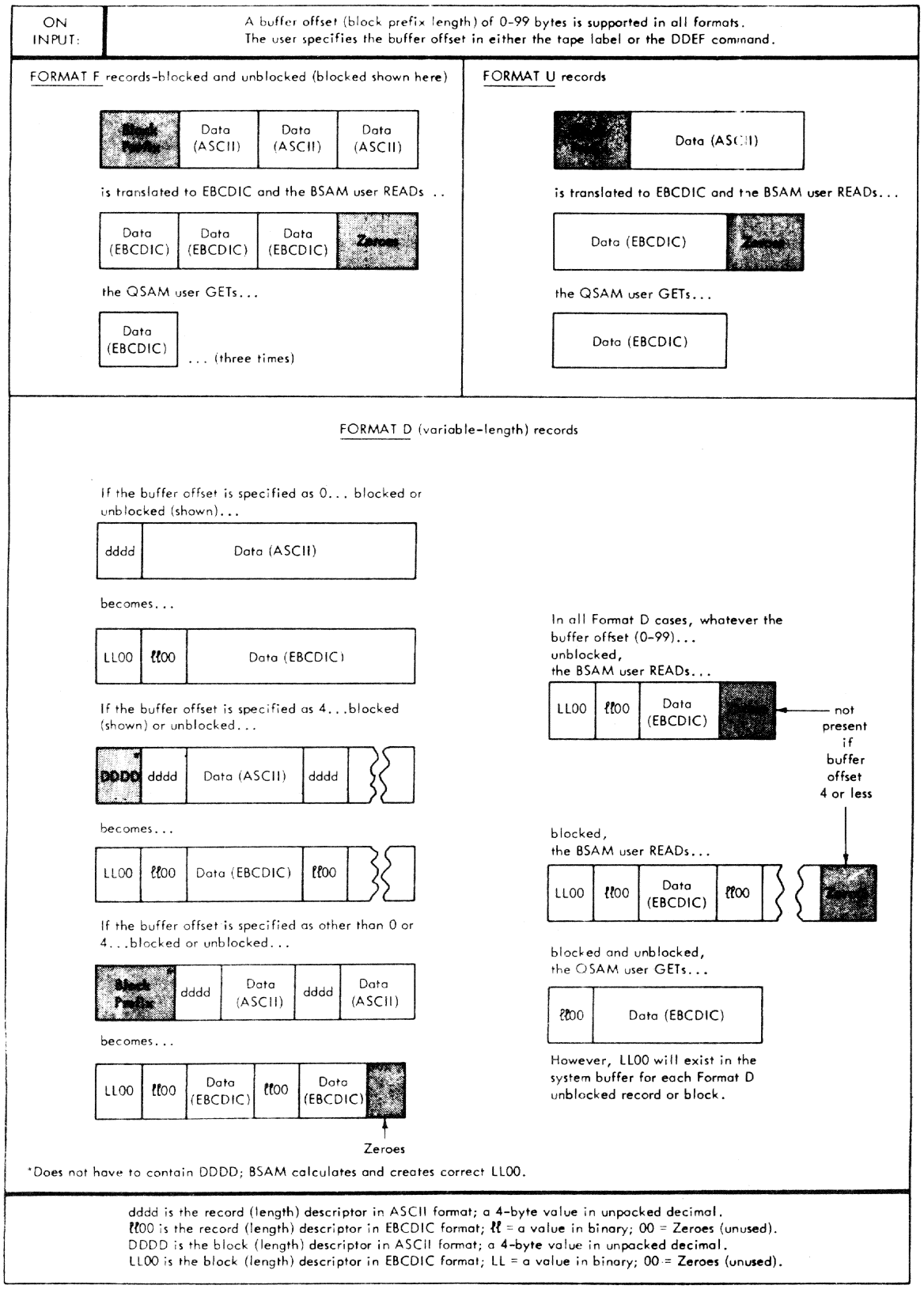


Figure 10. How TSS/360 Handles ASCII Record Input

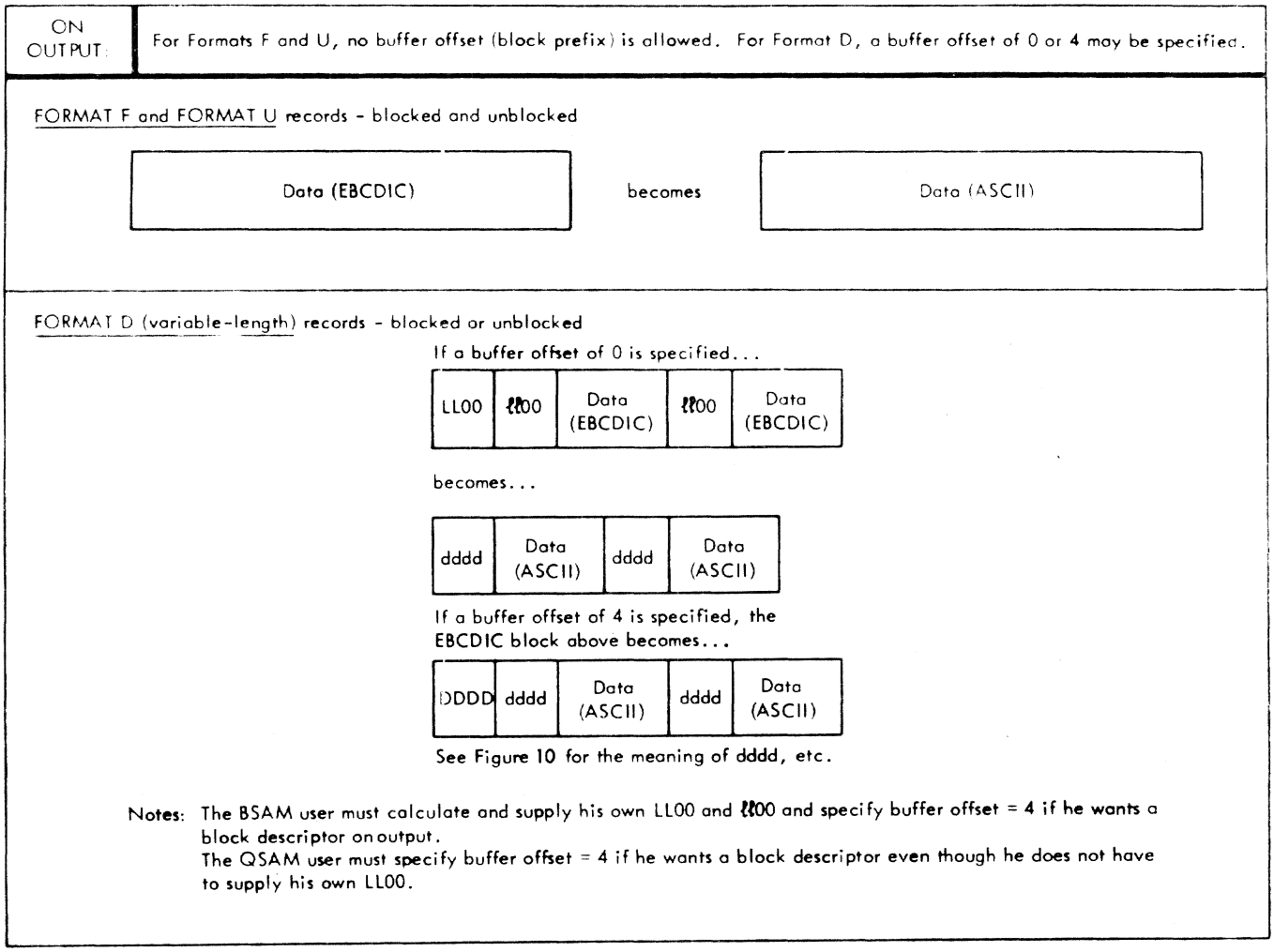
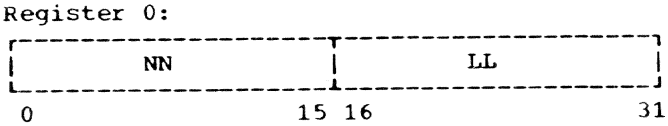


Figure 11. How TSS/360 Handles ASCII Record Output

Input: The following parameters are passed:

Register 0:



where NN - number of buffers requested.

LL - length of each buffer in the request.

Register 1 -- Address of DCB.

Data References: CHADCB, CHADEB, CHAISA, CHAVPS.

Modules Called: None.

Exits:
Normal -- Return to caller.

Error -- ABEND macro instruction.

Operation: The routine will ABEND if the DCBID is not valid. The routine also ABENDs if a buffer pool has been previously assigned, if the buffer length or block size exceeds 32,768 bytes, or if the number of buffers exceeds 255. GETPOOL inserts buffer length and number of buffers into DCBBUF and DCBBUN respectively. The routine sets on bits corresponding to DCBBUF and DCBBUN in DCBMSK if the DCB is open.

GETBUF Routine (CZCMA)

The GETBUF (get a buffer) routine finds an available buffer in a buffer pool and returns a pointer to it. When the GETBUF routine is entered for the first time, the buffer page list, which describes the location of all the buffers for this DCB, is built, the buffer pool is allocated, and the first available buffer is obtained. (See Chart HB.)

Attributes: Read-only, resident in virtual storage, closed routine, reentrant, public.

Entry Point: CZCMAG equated to SYSMAG -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB.

Data References: CHADCB, CHABPL, CHADEB, CHAISA, CHAVPS.

Modules Called:
VMA (CZCGA) -- Get virtual storage.

CKCLS (CEAQ4) -- Check protection class.

Exits:
Normal -- Return to calling program.

Error -- ABEND macro instruction.

Operation: GETBUF will ABEND if the DCBID is not valid, or if the DCB is not open.

If this is the first entry to GETBUF, a buffer page list must be built. GETBUF ABENDs if the buffer length is too large or if buffer length and blocksize are both zero. When only buffer length is zero, and access is for other than QSAM, and the device is direct access, the actual size of the buffer is computed by GETBUF by adding the key length to the blocksize. GETBUF determines which cutoff constant is to be used and calculates the total number of pages needed for the buffers, and the buffer page list.

GETMAIN is used to obtain the needed pages. GETBUF maintains the count of available buffers and enters the buffer addresses into the Buffer Page List.

FREEBUF Routine (CZCNA)

The FREEBUF (free a buffer) routine makes available a buffer which was previously obtained and made unavailable by the GETBUF routine. (See Chart HC.)

Attributes: Reentrant, resides in virtual storage, closed routine, read-only, public.

Entry Point: CZCNAF equated to SYSNAF -- Entered via type-1 linkage.

Input: The following parameters are passed:

Register 0 -- Address of buffer to be released.

Register 1 -- Address of DCB.

Data References: CHADCB, CHABPL, CHADEB, CHAVPS, CHAISA.

Modules Called: None.

Exits:
Normal -- Return to calling program.

Error -- ABEND macro instruction.

Operation: The addresses of buffers in CHABPL are searched to find a match for the address passed in register 0. When a match is found, the In-Use flag for that buffer is turned off in the CHABPL, and BPLNBF (number of free buffers) is incremented by one.

FREEBUF exits to ABEND if the DCB identification is not valid, if GETBUF has not been called, if the buffer to be freed is not in the pool, or if the buffer to be freed is already free.

FREEPOOL Routine (CZCNB)

The FREEPOOL (free a buffer pool) routine releases all virtual storage which was assigned to a DCB as a buffer pool. (See Chart HD.)

Attributes: Reentrant, resident in virtual storage, closed routine, read-only, public.

Entry Point: CZCNBC equated to SYSNBC -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB for the data set which last used pool.

Data References: CHADCB, CHABPL, CHADEB, CHAVPS, CHAISA.

Module Called: VMA (CZCGA) -- Free virtual storage.

Exits:
Normal -- Return to calling program.

Error -- ABEND macro instruction.

Operation: FREEPOOL ABENDs if the DCBID is not valid, or if the access is QSAM and the DCB is open.

Unless the Buffer Page List pointer is zero, the number of pages in the buffer pool is determined from BPLNPG, and FREEMAIN is called to release those pages.

FREEPOOL zeros the Buffer Page List pointer, and the buffer length and number of buffers fields in the DCB.

BSAM INTERNAL CONTROL ROUTINES

The internal control routines include the message writing, tape positioning, and volume serial field finding routines.

Tape Positioning Routine (CZCWP)

The Tape Positioning routine positions a tape volume to any of three positions. The positions are shown in Figure 12. (See Chart IA.)

Attributes: Reentrant, resides in ritual storage, subroutine, privileged.

Entry Point: CZCWP1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the SAM communication block. SCBPOS contains the code which indicates which tape position is desired.

Data References: CHADEB, CHASCB, CHASDA, CHATDT.

Modules Called:
Control (CZCRB) -- Tape positioning.

Message Writer (CZOWM) -- ABEND processing.

Build DEB (CZCWB) -- Update DEB after volume switch.

Bump (CZCAB) -- Mount next volume.

Volume Sequence Convert (CZCWV) -- Determine if there is another volume.

Exits:
Normal -- Return to the calling routine.

Error -- Call Message Writer to do ABEND processing.

Operation: Positioning is always relative to the tape marks on a volume. It is assumed that unlabeled volumes contain data sets separated by single tape marks, and that labeled volumes contain data sets with a single tape mark separating header labels and data, and a single tape mark preceding trailer labels.

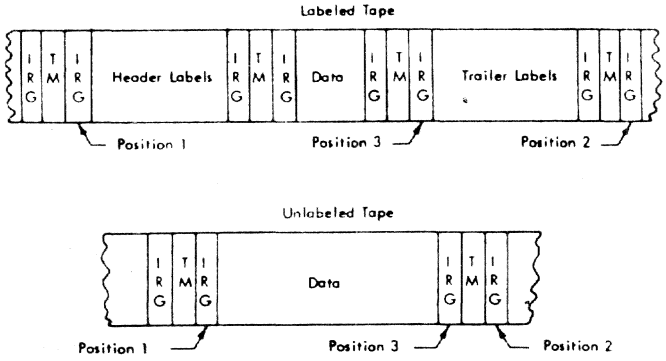


Figure 12. Tape Positions

The number of forward or backward spaces to be made must first be calculated. The calculation is made from the position code, the current tape mark count and the logical data set sequence number in the JFCB.

The present position on the tape volume is known from the tape mark count which indicates how many tape marks are behind the present position. For example, in Figure 13 there are unlabeled data sets on a tape volume (the current positioning is indicated); the tape mark count at this position would be two.

Calculating the Position to Which the Tape is to be Moved: A temporary data area, TFN, is set to the relative physical sequence number of the current data set of the current volume. This is calculated by subtracting the first logical file (data set) sequence number on the mounted volume (TDTLFN - 1) from the logical file sequence number (TDTPSQ) of the data set which the user requested (TDTFSQ - TDTLFN + 1). For example, in Figure 13 the TDTFSQ contains a 2 if this is the only volume, indicating that current positioning is somewhere within the second data set on the volume.

It should be noted now that one tape mark follows every data set on an unlabeled tape volume. Similarly, since there are three tape marks associated with each data set on labeled tape, multiplying the physical sequence number of the current data set (TFN) by three, then subtracting one, yields the exact number of tape marks preceding the first TM of the current data set.

Now that the number of tape marks associated with the data set on which positioning is to occur is known, it only remains to determine how many tape marks beyond the previous data set are needed to find the desired new position in terms of tape marks.

Looking at Figure 12, it is clear that for labeled tape: zero additional tape marks yields position 1, three additional tape marks yields position 2, or two additional tape marks yields position 3. For unlabeled tape, zero additional tape marks yields position one, while one additional tape mark yields position 2. Position 3

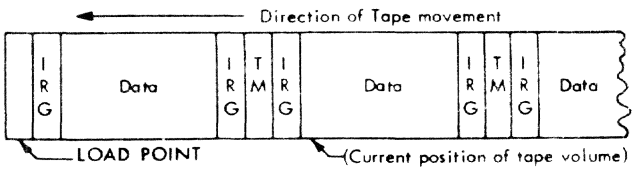


Figure 13. Data Positioning

cannot be found directly now because files (tape marks delimit files) are being counted in the forward tape direction. A backspace file is necessary to obtain position 3 on nonlabeled tapes after forward space files are completed.

The appropriate number of additional tape marks is put into a temporary data area TFA.

Since all information necessary to find the new position is known in terms of tape marks, the calculation can be made:

labeled tape TFS = (TFN-1)x3+TFA

or

unlabeled tape TFS = (TFN-1)+TFA

TFS is a temporary data area which contains the desired new position in terms of number of tape marks from the beginning of the tape volume.

The difference between TFS, the desired position, and SDATAP, the present position, is determined by subtraction. The result is the number of files (delimited by tape marks) to be forward or backward spaced. The direction to go is indicated by the sign of the above difference.

The Control routine is used to forward or backward space the required number of files.

The following situation may occur when skipping files forward or backward. The solution is the same for either case. As in Figure 14 there is a requirement for forward spacing.

When forward spacing to the correct position, the following procedure is followed if Tape Positioning was called by Tape Open:

Labeled tapes - A forward space record is done to check for a tape mark, which would indicate the end of the tape. If a tape mark is encountered and another tape is not specified, an ABEND results. If a tape mark is not encountered, three forward space files are done to position to the beginning of the next data set. This procedure is followed until the correct file (data set) is reached. Then another forward space record is done to check for an end-of-tape condition. If a tape mark is encountered, a check is made for another volume. If there is not another volume and the user is open for input and reading forward, his task is ended via ABEND. Otherwise, the tape is positioned correctly.

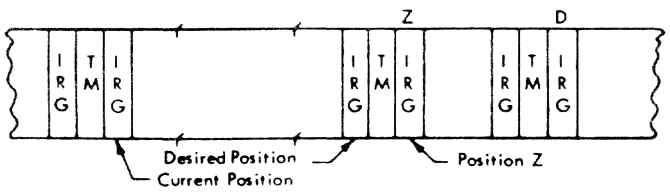


Figure 14. Skipping Files on Tape

Unlabeled tapes - The same procedure is followed as for labeled tapes, except only one forward space file is done after the forward space record.

Multivolume Check - If the JFCB indicates another volume on an end-of-volume condition, it is mounted, the DEB is updated, and the tape positioning values are updated for the newly-mounted volume. In addition, the logical file sequence number is placed in the volume entry (TDTFSQ) of the TDT.

Forward motion of the tape can only result in position Z or D because spacing by files (delimited by tape marks) causes the tape to stop in the interrecord gap after the skipped file. Therefore, to reach the desired position in Figure 14, files are skipped forward to reach position Z, and then there is a backward skip of one file position to the desired position.

Volume Sequence Convert Routine (CZCWV)

The Volume Sequence Convert routine is called to determine the address of a volume serial field within the correct job file control block (JFCB) within the task definition table, based on a Relative Volume Sequence Number (RVS) stored in the SAM communication block (CHASCB). (See Chart IB.)

Attributes: Reentrant, nonrecursive, resident in virtual storage, privileged.

Entry Point: CZCWV1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the SAM communication block.

Data Reference: CHASCB.

Modules Called: None.

Exits:
Normal -- Register 15 contains one of the following return codes:

'00' Address of the volume serial field requested is in SCBVCA and the field is valid.

'04' SCBVCA contains the address of a null volume serial field, or a null chain field.

'08' Volume serial field considered out of range.

Error -- None.

Operation: The calling program stores the address of a volume serial field within the JFCB in the CHASCB, and places a return code in general register 15 indicating the result of the search.

The return code setting indicates to the calling program whether the volume serial field has been located, and if it has, its address (stored in CHASCB). Otherwise, the return code indicates whether the RVS stored in the CHASCB points to a null volume serial field and the address of this field is stored in the CHASCB; or whether the RVS points to a volume serial field which would begin the next chain of volume serial entries in the JFCB (in which case the CHASCB contains a pointer to the null chain field which will be used to point to the address of the field requested by the RVS). The return code, on the other hand, may indicate that the RVS stored in the CHASCB points to a volume serial field which does not exist in the current JFCB (in which case a pointer to the terminating null chain field is stored in the CHASCB), or the return code may indicate that the RVS was zero upon entry, and the address of the volume serial field pointed to by RVS has been set to zero.

Note: The first volume serial field in the JFCB is assumed to be 1.

The RVS stored in the CHASCB is not changed by VOLCVT.

Message Writer Routine (CZCWM)

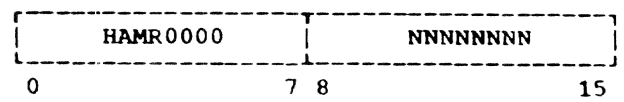
The Message Writer provides for all the message handling required by the SAM Open, End-Of-Volume, and Close Routines. This module contains all message text. It selects the proper method of transmitting the message to the user or to the operator. Some standard responses are processed and returned to the calling SAM modules in the form of return codes in general register 15.

The Message Writer also handles all ABEND processing for BSAM. (See Chart IC.)

Attributes: Reentrant, nonrecursive, resident in virtual storage, privileged.

Entry Point: CZCWM1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of CHASCB. The two-byte SCBMSG field of CHASCB is set as follows:



where H = 0 No header required

1 Message requires header

A = 0 ABEND

1 Prompting message

M = 0 User message

1 Operator message

R = 0 No response

1 Response required

and byte 1 is a message ID consisting of a binary number. There is a unique number for each message.

The SCBABN field contains the SAM ABEND code, if the call is for an ABEND message.

Data References: CHASCB, CHATDT, CHASDA, CHADEB, CHALB1.

Modules Called:
WTO (CZABQ) -- Write message to operator.

Gate (CZAAB) -- Print message on terminal or SYSOUT device.

ABEND (CZACP) -- Abnormal termination of a task.

Exits:
Normal -- Return to caller.

Error -- ABEND.

Operation: This routine uses the message code placed in the CHASCB as an index to locate the desired entry in the message control table (MSGTS). A message is then formed in a buffer area in the PSECT of this routine. All messages begin with a prefix, contain a header if the message is for a user, and end with the message text.

The prefix is built from information in the CHASCB. It contains the module name of the caller, the abend code, and the message code. This uniquely identifies each message.

When the message is for a user, a header line is added. This provides the DDNAME

and DSNAME to identify the data set being processed.

The phrase or phrases forming the message text are concatenated with the prefix and header in the buffer area. If necessary, a modification routine is invoked to complete variable fields in the text. The completed message is then transmitted as indicated by flags in the message control table.

If the message control table flags indicate that a response is required, it is returned to the user by placing a pointer to the response buffer in the CHASCB. If possible, the response will be interpreted and a return code set to facilitate testing by the caller. When the expected response is not obtained, a retry message is transmitted with the same prefix used for the original message. This cycle is then repeated until the proper reply is obtained.

The Message Writer routine is called to do ABEND processing when one of the following conditions occurs:

1. The message code received from the caller indicates an entry outside the range of the message control table or an entry within the table that is no longer active.

2. The message is too long to be concatenated in the buffer available.

3. The modification or response routine is not available when indicated.

4. No SDAT pointer was available when required to complete the operator message.

The Message Control Table: The message control table contains one entry of two words for each message. Each entry, as illustrated in Figure 15, has the following format:

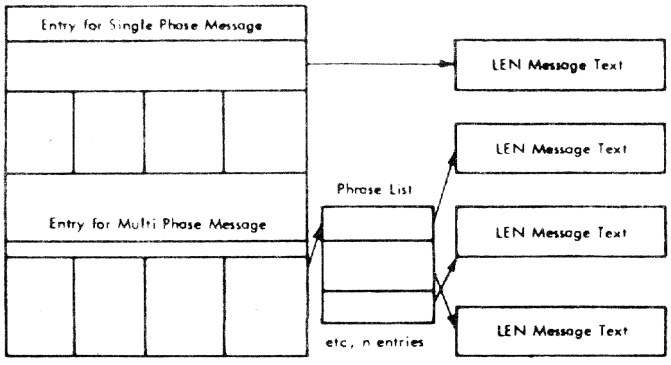


Figure 15. Entry for Single/Multiple Phase Message

DC A(MSG)	Pointer to message text if PP is zero. Pointer to Phrase List of PP entries if PP is not zero.
DC 'MM'	Code for Modification routine (00=none).
DC 'RR'	Code for Response routine (00=none).
DC 'PP'	Number of entries in Phrase List.
DC 'FF'	Flags.

	0	1	2	3	4	5	6	7
Do not print header.	0	X	X	X	X	X	X	X
Print header.	1	X	X	X	X	X	X	X
Abend text.	X	0	X	X	X	X	X	X
Prompting Message text.	X	1	X	X	X	X	X	X
Message for User.	X	X	0	X	X	X	X	X
Message for operator.	X	X	1	X	X	X	X	X
No response required.	X	X	X	0	X	X	X	X
Response required.	X	X	X	1	X	X	X	X

Find Records per Track Routine (CZCRQ)

The Find Records Per Track routine computes the number of records on a track of a direct access device. For a data set of unknown or varying record length, an IORCB is set up to read the entire track in order to find the number of the last record. (See Chart ID.)

Entry Point: CZCRQA -- Entered via Type-1 linkage.

Input: Register 1 contains the address of the DCB. DCBRDB contains the address of the track concerned.

Modules Called: None. The AWAIT and IOCAL SVCS are issued.

Exits:

Return Code	Condition
00	Normal return.
04	I/O error other than defective track or 'No Record Found'.

Operation: For fixed, standard format records, the record count is computed and stored in the DCBRDR field. A return to the calling routine is then effected.

For record formats other than fixed, or fixed but not standard, an IORCB is built, and an IOCAL issued, which returns the record count to a data area in the PSECT. The count is then stored in the DCBRDR field of CHADCB.

This count is also stored in the DCBRDR field if an error return of 'No Record Found' is detected after the IOCAL SVC.

If the IOCAL return shows that the track was defective, the alternate Seek address is moved into the IORCB and the IOCAL issued again.

Any other error return from IOCAL causes Find Records per Track to return a code of X'04' in register 15.

RELFUL Routine (CZCRR)

This routine converts a relative direct access device address - of the form TTRZ, to a full address - of the form RMBBCCHH, where R is the record number, M is the extent number, CC is the cylinder number, and HH is the head (track) number. (See Chart IE.)

Entry Point: CZCRRA (SYSRRA equated) -- via type-1 linkage.

Input: Register 1 contains the address of the DCB. DCBRDT contains the relative address to be converted.

Modules Called: None.

Exits:

Return Code	Condition
00	Normal return.
04	Track number is outside data set.

Operation: The extent number (M) and record number are moved in from the DEB. The track number is computed and if it is on the first cylinder of the extent, the cylinder and track (head) are placed directly into the DCB and a return to the calling modules effected.

If the track is not on the first cylinder, the cylinder count is updated until the track is found, and the cylinder and track are computed and stored in the DCB. A return to the calling module is effected.

FULREL Routine (CZCRS)

FULREL converts a full DA device address in the form RMBBCCHH to a relative address in the form TTRZ. R is the record number, M is the extent number, BB is the bin number, CC is the cylinder number, and HH is the head (track) number. TTRZ represents the relative track and record numbers for the current volume. (See Chart IF.)

Entry Point: CZCRSA (SYSRRA equated) -- via type-1 linkage.

Input: Register 1 contains the address of the DCB.

Modules Called: None.

Exits: Return to the calling routine. No return code.

Operation: The address is converted to the TTRZ form where the R of TTRZ is set to the R of RMBBCCHH minus one, the Z is set to zero, and the track is computed by adding the number of tracks in the extents. The remaining fields of DCBRD are set to zero.

PART II

VIRTUAL ACCESS METHOD (VAM)

The data management facility discussed in this section is the virtual access method (VAM). There are three organizational methods: virtual sequential, virtual indexed sequential, and virtual partitioned, which are designed for use with TSS/360. All incoming and outgoing data processed by the virtual access methods is organized in units of pages which are 4096 bytes in length and stored on direct access devices.

TSS/360 VAM is designed to minimize the number of virtual storage pages associated with an open data set. VAM brings into virtual storage only those data set pages currently needed for user operation.

In this section, the implementation of VAM is presented using seven subdivisions:

- A general description of VAM including its unique characteristics and its special capabilities.
- A description of error recovery techniques used for VAM data sets.
- A description of the volume format and the manner in which data sets are maintained on such a volume.
- Data set sharing - a detailed discussion on one of VAM's more important facilities.
- Open and Close processing - that part of VAM processing common to all the access methods.
- The virtual sequential access method.
- The virtual indexed sequential access method.
- The virtual partitioned access method.

Where practical, individual routines will be presented immediately following the particular facility they support. The first of the above subdivisions, a general description of the virtual access methods, will occupy the remainder of this section.

VIRTUAL DATA SET ORGANIZATION

As metioned above, the data sets accessed by the virtual access methods all reside on direct access storage devices. The devices supported by TSS/360 are the 2311 and 2314 direct access disk storage devices. Each device is preformatted, in a manner which will be described later, in units of 4096 bytes called pages. Although records in a data set may occupy less than a page, exactly one page, or more than one page, the access method deals in terms of page units.

The general philosophy of the virtual access methods is closely tied to the concept of relocation exceptions. When a user issues a request to get or read a record, the virtual access method merely adds the page or pages containing the record to the requesting task's page tables. This operation is performed by the Movepage routine (CZCOC). That is, the record is read into the task's virtual storage. When the task in execution makes a reference to the record, a page relocation hardware interruption occurs. This interruption causes the resident supervisor to read the referenced page into main storage by means of its paging mechanism.

A put or write operation, on the other hand, results in the actual writing out of the record as well as the updating of the page tables. The writing operation is performed by the Movepage routine (CZCOC) which issues the PGOUT SVC. This SVC is also serviced by the resident supervisor and performs the actual writing operation.

Movepage Routine (CZCOC)

Movepage is called by OPENVAM (CZCOA), CLOSEVAM (CZCOB), VSAM Get (CZCOR), VSAM Put (CZCOS), SETL (CZCOT), FLUSHBUF (CZCOT), Getpage (CZCPI), and VISAM Close (CZCQA) to perform input or output and to control the use of shared pages by setting and releasing interlocks. (See Chart JA.)

Attributes: Read-only, public, privileged.

Entry Point: CZCOC1 - Entered via type-1 or type-2 linkage.

Input: Register 1 contains a pointer to a two-word parameter list:

Word 1 -- Address of DCB.

Word 2 -- Address of first page of request.

Field DCBN contains the relative page number of the first data set page of the request.

Field DCBM contains the total number of contiguous pages involved in the request.

Field DCBOP contains the type of operation requested as follows:

'80' Input request.

'20' Input request with exclusive read.

'10' Output request.

'02' Release read lock.

'01' Release write lock.

Modules Called:
GETNUMBR (CZCO01) -- Converts the page number of a partitioned data set relative to a member to the relative page number with respect to the start of the data set if part of the member has been moved to an overflow page.

Interlock (CZCOH1) -- Interlocks the external page entry of a shared data page or interlocks the entire RESTBL.

Release Interlock (CZCOI1) -- Releases the above interlocks.

PGOUT (CEAA1) -- Causes the pages of an output request to be written to external storage.

SETXP (CEAH7) -- Adds the pages of an input request to the task's external page table so that the supervisor can bring them into main storage when a reference to them causes a relocation exception.

RELEXPG (CZCEN1) -- Releases pages assigned to the data set but not in use.

FINDEXPG (CZCEL1) -- Assigns new pages to the data set.

Exits:
Normal -- RETURN macro instruction.

Error --

1. Return with a code of X'04' in general register 15 if an attempt to perform an exclusive read failed because the page was already locked.

2. ABEND is called under the following conditions:

   - An unrecoverable error occurs during the PGOUT operation.
   - The DCB in the parameter list is not the same as the one in the DCB header.
   - Illegal data set organization exists.
   - An output operation is requested on a data set opened for input. This is not the case if the output operation is to output the POD of a partitioned data set.
   - The request is for input from beyond the data set limits.
   - The operation requested is invalid.
   - The DSORG is indexed, and DCBM ≠ 1.
   - An error return is received from GETNUMBR.
   - An illegal buffer is passed to Movepage for VI organized data sets.
   - The PGOUT buffer is not data or overflow.

Operation: On entry, registers are saved and base registers are established in conformance with linkage conventions. If the data set is shared, the RESTBL is interlocked. If the address of the DCB is not in the DCB header, or if the data set is not partitioned and the request is for output when the DCB was opened for input only, a call is made to ABEND. This latter operation is valid for partitioned data sets since the request may be from CLOSEVAM and be a request to write out the POD.

If the data set is partitioned, the relative page number must be determined. The user reference is to a relative page number based on the start of the member; this relative page number must be converted to one based on the start of the data set. Where part of the member has been relocated to an overflow page, GETNUMBR is called to make the relative page number determination.

After this calculation or if the data set was not partitioned, the request is examined further. If the request was for an output operation on a partitioned data set opened for input, and the request was not to output the POD, ABEND is called. If this error condition does not exist, processing continues. The extent of the requested operation is tested to see if the pages involved are all within the limits of the pages currently assigned to the data set. If the data set limits will be exceeded by the request, ABEND is called. If all is in order, the processing continues. From this point on, the processing differs according to the type of request.

Output Operation: In general, the output of data set pages is accomplished by means of the PGOUT SVC. The processing in Movepage consists of building a parameter list and issuing the SVC. The parameter list

for PGOUT contains the first page involved, the number of pages, and the virtual storage address of the first external page entry. The page entries must be converted from the format used in the RESTBL (relative volume number-relative external page number), to a format suitable to the PGOUT processor (symbolic device address-external page number). This conversion is accomplished by directly indexing into the volume table and appending the SDA to the given external page number. This parameter is then placed in the PGOUT list. This process is repeated, page by page, until all pages involved have been placed in the parameter list or until the maximum of 8 entries have been placed in the list. When either of these conditions is met, the PGOUT SVC is issued and the transfer of pages is accomplished. On return from PGOUT, the data set is tested to see if it is duplexed (see duplexing later in this section). If it is, the same parameter list is used to write the pages to the secondary copy of the data set. When both copies have been written out, the procedure is repeated, if necessary, for each succeeding group of 8 pages until the request has been satisfied.

Once this has been done the exit procedure is entered. This entails the release of RESTBL and page interlocks set on shared data set pages and RESTBLs, and a return to the calling routine.

Non-output Operation: Operations which are not output may be simple input operations on a nonshared data set, input from a shared data set, a read exclusive request for a shared data set, or a request to release a read or write interlock on a shared data set page. If none of these operations is indicated, a call is made to ABEND.

If the request is for a simple input operation on a nonshared data set, the virtual storage address of the first page in the request is computed and placed in the parameter list for SETXP. As with the parameter list constructed for PGOUT, these page entries must be converted to a form acceptable to the SETXP processor. The conversion process is identical. Page entries are converted one by one and placed in the parameter list until the list maximum of 64 entries is completed or until the requested page entries are all in the list. The SETXP SVC is then issued. This action results in the placement of the pages involved in the external page table of the task. Any future reference to the pages will result in a relocation exception which can now be processed by the resident supervisor. If there are more than 64 pages involved in the requests, this process is repeated until all requested pages have been placed in the external page table.

If this input request was not received from Open, the number of checked out pages is updated. After that or if the request was from Open, control is returned to the calling routine.

The processing of input requests for shared data sets is basically the same as for nonshared data sets. The only difference is that page level interlocks must be set and released in certain operations. If the data set organization is other than VISAM, no interlocks must be set, so the above input processing is entered. The parameter list for the SETXP SVC is established and the SVC is issued. If the data set is VISAM but the request is for a directory page the same processing occurs.

When the request is for a data page of a VISAM data set, one of two interlocks must be set on all pages involed in the request. For data sets opened for input or for operations other than read exclusive, a read interlock is set on all pages; for a read exclusive request, a write interlock is set. The significance of these interlocks and the manner in which they are imposed is discussed in the section on sharing.

Once the type of interlock has been determined, the pages are examined to see if an interlock is already set. This occurs when an exclusive read request is made for a page which is already in use. In such a case, a return is made to the caller with a return code of '04' to indicate that the page is not currently available. If the page is not already locked, a call is made to Interlock to impose the proper interlock on all pages. This interlock is not imposed if the call to Movepage is for a read request from the dynamic loader. Following the setting of appropriate interlocks, the parameter list is built and the SETXP SVC is issued as for nonshared input requests.

After the SVC has been processed and control has returned to Movepage, the number of pages checked out is updated for all requests except those from Open, the RESTBL interlock is released and control is returned to the calling routine.

The remaining function performed by Movepage is the release of the page level interlocks set for input requests on the data pages of a VISAM data set. The interlocks set can be either read or write. The type of interlock is determined and passed to the Release Interlock routine as a parameter. Once the interlocks on the pages have been released, the interlock on the

RESTBL is released by again calling Release Interlock, and control is returned to the calling routine.

THE ACCESS METHODS

Associated with each of the organizational methods is an access method by which the user may access records in his data set. These access methods are: the virtual sequential access method (VSAM), the virtual indexed sequential access method (VISAM), and the virtual partitioned access method (VPAM). This last facility is not an access method in the normal sense of the term. VPAM contains no routines for the actual reading or writing of records. A virtual partitioned data set is a collection of other data sets which a user has combined for ease of reference. These subsidiary data sets are called members and each member is itself organized as a virtual sequential or virtual indexed sequential data set. It is by means of the other access methods that the records of a member are actually read into the task's virtual storage.

VPAM provides the additional control to perform the following functions on members:

- To create or add to a virtual partitioned data set.
- To prepare any member of a virtual partitioned data set for processing.
- To add new members to, or delete existing members from an existing data set.
- To update existing members in place.

The virtual sequential access method provides the user with access to records that are located in logically sequential locations in his virtual storage. Because of the nature of the virtual storage concept, these records may or may not be in physically sequential locations in main or external storage, but they may be conceived of as being sequentially organized for processing purposes. Tables maintained by the data management routines and by the resident supervisor make this type of processing possible. The manner in which this is done is described in detail in the section on the virtual sequential access method.

The virtual sequential access method (VSAM) processes virtual sequential data sets and virtual sequential members of partitioned data sets. It can be used for any of the following functions:

- To create or extend a virtual sequential data set or virtual sequential member of a partitioned data set.
- To delete all records in an existing data set or member, from a specified record to the end of the data set.
- To retrieve the logical records of the data set or member in a sequential manner.
- To update an existing record of the data set or member in place.

The virtual indexed sequential access method provides the means by which a user can access records in a virtual indexed sequential data set. Such data sets contain records which are not sequentially located in virtual storage. Each record is associated with a key which is contained in the record in storage. The lowest record key in each data page, except the first data page, of a virtual indexed sequential data set is also entered in a directory which is associated with each such data set. By referencing the keys associated with the records in a sequential manner, the user may process his data set as a sequential data set. Optionally he may access records in a nonsequential manner, selecting the records he wishes in any order by referencing the appropriate keys. This processing is detailed in the section on the virtual indexed sequential access method.

The virtual indexed sequential access method (VISAM) processes virtual indexed sequential data sets or indexed sequential members of partitioned data sets. It can be used for any of the following functions:

- To create a virtual indexed sequential data set or member in a sequential manner.
- To retrieve the logical records of the data set or member in a sequential or nonsequential manner.
- To update records in a sequential or nonsequential manner.
- To insert records in logical sequence within the data set or member.
- To delete selected records from the data set or member.

FACILITIES PROVIDED BY VAM

In addition to the restriction that they reside only on direct access devices, VAM data sets are characterized by two facilities that they provide for the user.

The first VAM facility is the sharing of data sets. A user may elect to share his data set with other users. When he does

this he is known as the data set owner and any user who shares the data set with him is known as the sharer. The extent to which the sharer may use the data set is determined by the owner when he permits the sharing. He may permit read only access, read-write access, or unlimited access.

The sharing of data sets necessitates the use of interlocks which prevent two or more users from simultaneously accessing the data set. These interlocks and their use, as well as the rules for sharing data sets and the routines involved in the sharing process, are discussed in the section on sharing.

The second of these facilities is duplexing. This facility provides the user with an error recovery capability by allowing him to maintain two identical copies of his data set. The user specifies this option by opening his data set with the DUPOPEN macro instruction rather than with the OPEN macro instruction. The net effect of this is to link two identical DCBs together and to flag his data set RESTBL as duplexed. The duplicate copy is updated by the Movepage routine as previously described and, other than the DUPOPEN macro instruction, the user processes his data set as he normally would and does not concern himself with the duplexing operation.

VAM ERROR RECOVERY TECHNIQUES

Two routines are used by data management to attempt error recovery, or to allow the user to make the decision to attempt recovery, rather than cause an ABEND to destroy the user's task.

The Virtual Memory Input Error Recovery (VMIER) routine is invoked when an error is encountered on an input operation taking taking place on a direct access device. If the user has maintained a duplexed data set, the secondary copy is used to replace the error page in a newly assigned external location of the primary data set, and processing continues.

The VAM Data Management Error Processing (VDMEP) routine is designed to process most errors occurring while manipulating data sets. The data set in question is closed out, appropriate diagnostics are generated, and control returned to the user.

VMIER and VDMEP are not to be confused with VMER (Virtual Memory Error Recording) and VMSDR (Virtual Memory Statistical Data Recording); these modules are called by certain access methods posting routines and are described in System Service Routines, GY28-2018.

VMIER Routine (CZCEI)

The Virtual Memory Input Error Recovery routine is a public, read-only, privileged, system routine which is called by the Task Monitor to attempt recovery from an input error occurring on a direct access device. (See Chart JB.)

Entry Point: CZCEI1 - Entered via type-1 linkage.

Input: None. The ISA must contain:

ISAORV -- Virtual storage address into which the error page was to have been read.

ISAORE -- External page address from which the error page was to have been read.

Modules Called:

SETXP (CEAH7) -- Sets the task's external page table to point to the secondary copy of a duplexed data set so that a good copy of the error page may be obtained.

PGOUT (CEAA1) -- Causes the secondary copy of the error page to be written to a new primary copy location.

FINDEXPG (CZCEL) -- Assigns a new data page to the data set. This page will replace the error page and the secondary copy of the copy of the error page will be written to it.

Interlock (CZCOH) -- Places an interlock on the RESTBL and the external page of a shared data set.

Release Interlock (CZCOI) -- Releases the aboye interlocks when appropriate.

WTL (CZABQ) -- Writes a message to the system log when a PAT page is in error.

WTO (CZABQ) -- Informs the operator of a successful recovery.

GATWR (CZATC) -- Writes a message to the task's SYSOUT when a PAT page of a private volume is in error.

ABEND (CZACP) -- Abnormally terminates a task under certain error conditions.

DSCB/CAT Recovery (CZUFX) -- If the error page is in USERCAT or SYSCAT data sets.

Exits:

Normal -- Return to the task monitor.

Error -- ABEND is called under the following conditions:

- The error occurred on an auxiliary paging volume.
- An unrecoverable error occurred on a data page (data set nor duplexed).
- The device on which the error occurred was not a direct access device.
- The error page is outside the limits of the device.
- No matching external page entry is found in any open data set.
- An error page is found to be in the secondary copy of a duplexed data set.
- The error page is found to have been previously marked in error.

Note that the third, fourth, and seventh ABEND conditions listed are impossible assuming a properly functioning supervisor. They should be taken to mean bad parameters.

Operation: VMIER provides the processing which enables the system to make use of the duplexing feature in virtual organization data sets. When an input error occurs on a primary data page for such a data set, the secondary copy of the data page is read and rewritten to a newly assigned primary data page location in external storage.

When first entered, VMIER saves the input registers and establishes base registers for the CSECT, PSECT, and SDAT. It next locks the SDAT by means of the Test and Set instruction and locates the SDAT entry for the device on which the error occurred. If the error occurred on a device other than direct access or on an auxiliary paging device a call is made to ABEND. The same call is made if the error page is found to be outside the limits of the device.

The error page is next classified as being a PAT page, a DSCB page, or a data page. If it is a PAT page, the page is locked and all the page entries on the page are set unavailable for assignment, by setting the number of users currently assigned to the maximum of 127. This process involves a call to PGOUT to write out the updated PAT page. Those pages represented by the PAT page in error, are shown unavailable for assignment in the SDAT entry for the device and the PAT page lock is released. WTL (CZABQ) is called to make a note in the system log concerning the error. If the volume is private a message is written to the task's SYSOUT via GATWR (CZATC). After this, or if the volume was public, a return is made to the task monitor.

If the error page is not a PAT page, the PAT page is locked and the error page is examined. If the page is already marked in error, an attempt has been made to read an error page and ABEND is called (since other VAM routines, especially OPENVAM, will previously have relocated all known error pages). If this is not the case, the page is examined to determine if it is a DSCB page or a data page.

For a DSCB page, the entry in the PAT is marked in error and the error page is rewritten to its original location. This should permit the page to be input correctly (that is, without parity errors) on subsequent references and prevent superfluous calls to VMIER. The resulting data errors will be detected via checksum validation by those VAM routines which read or write DSCB pages that is, VAMOPEN, ADDDSCB, WRITDSCB, CATVAM, etc. Both of these operations involve calls to the PGOUT processor. If the DSCB page is determined to be for the system catalog data set (SYSCAT), a call is made to the DSCB/Catalog Recovery routine to rebuild the catalog; the error page is not rewritten to its original location. Following this, the SDAT lock is released and control is returned to the task monitor.

When the error page is found to be neither PAT nor DSCB, the chain of JFCBs in the TDT is searched in an attempt to locate the data set which contains the error page. This is done by scanning the RESTBLs of all open (that is, currently in use) VAM (on direct access) data sets for a match. If the entire chain of JFCBs is checked without finding a match, ABEND is called since this indicates a possible VAM malfunction (no page should be input which is not part of some VAM data set). During the search all RESTBLs for shared data sets are locked while being checked and unlocked afterwards. If the secondary RESTBL is located first, the primary RESTBL is used to obtain header information needed for the scan (as this information is identical for both RESTBLs but is maintained only in the primary RESTBL header). Both primary and secondary RESTBLs will eventually be searched for the proper external page entry. If the RESTBL does not contain the page, the next JFCB in the chain is retrieved and the search continues.

Once the error page entry has been located in a RESTBL, its entry in the PAT is set to indicate the page is in error. Since recovery is possible only for duplexed data sets, if the set is nonduplexed, a call is made to ABEND. This is also done if the data set is duplexed but the error occurred on the secondary copy since the secondary copy will only be read by VMIER when trying to recover from an error to the

failing primary copy. Whether duplexed or not, the error page is checked to see if it is in the USERCAT or SYSCAT data sets; if so, a call is made to CZUFX to rebuild the catalog and return is made to the task monitor. If the error page is in the primary copy, recovery is possible so the secondary copy address is set in the external page table via a SETXP call. A call is then made to FINDEXPG to assign a new page to contain the primary copy, that is, the error page replacement in both the RESTBL and the DSCB for the data set. When the new page has been assigned, PGOUT is called to write the secondary copy to the new primary location. Write to Operator is then called to inform the operator of a bad page on one of the system packs and, incidently, that VMIER was invoked successfully; all interlocks are then released and a return is made to the task monitor.

VDMEP Routine (CZCQK)

The VAM Data Management Error Processing (VDMEP) routine processes all the errors detected while mainipulating a data set. VDMEP will close the data set which caused the error, release the interlocks set, and transmit diagnostic messages to the user's SYSOUT. VDMEP is called by the VDMER macro expansion or by ABEND (CZACP).

If the task is conversational, control is returned to the terminal; otherwise the task is deleted. (See Chart JC.)

Entry Points:

CZCQK1 -- Entered from expansion of the VDMER macro.

CZCQK2 -- Entered from CZACP (ABEND) when ABEND receives a recursive call while processing a VDMEP request.

CZCQK3 -- Entered from CZACP (ABEND) when ABEND has successfully completed a VDMEP request.

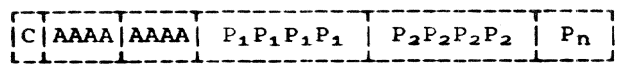
Input:

For entry at CZCQK2 and CZCQK3, there are no parameters passed.

For entry at CZCQK1, register 1 contains the address of the following parameter list:

Word 1 -- Address of DCB for the data set in error.

Word 2 -- Pointer to an 8-character Message ID, preceded by a 1-byte count of pointers to variable data, and followed by the pointers.



C = 1-byte count of pointers. (May be zero.)

A = 8 character message ID. This doubleword is actually addressed by word 2 of the parameter list.

P_1, P_2,...P_n = 4-byte pointers to variable data, if any.

Word 3 -- Pointer to a 2-byte field:

Byte 1	Condition
'10'	EODAD or SYNAD condition
'20'	Clear Last Operation flag

Byte 2	Condition
'0A'	Called by one of the 'OPEN' modules - CZCOA, CZCPZ, CZCOP
'0C'	SDST error in CZCOA
'0E'	Non-VAM data set in CZCOA

Modules Called:

VAM ABEND Interlock Release (CZCQQI) -- Release interlocks.

FREEMAIN (CZCGA3) -- Free virtual storage.

FINDJFCB (CZAEB1) -- Get JFCB address.

RELEXPG (CZCEN1) -- Release DSCB slot.

DELCAT (CZCFD1) -- Delete catalog entry.

Search SDST (CZCQE1) -- Close SDST entry.

Interlock (CZCOH1) -- Set RESTBL lock.

Close Common (CZCLBC) -- Close data set.

Release Interlock (CZCOI1) -- Release RESTBL lock.

Disconnect (CZCGA8) -- Disconnect from shared virtual storage.

Prompt (CZATJ1) -- Communicate with user terminal.

Stow (CZCOK1) -- Add or replace VP member.

XWTO (CZABQ1) -- Communicate with the system operator.

DUPCLOSE (CZCEZ1) -- Close duplexed data set.

CZAWA1 -- BULKIO ABEND recovery.

ABEND (CZACP1) -- Abnormal task termination.

ABEND (CZACP3) -- Successful VDMEP completion.

Operation: This text is keyed to the flowchart for CZCQK (Chart JC) and a reference by label will be included for each area. This routine will assemble appropriate diagnostics to the user SYSOUT informing the user of the original error which caused the call to VDMEP, the data set name of the data set for which the error was detected, and any action taken on the data set.

For entry at CZCQK1: After standard linkage is performed, a check will immediately be made to determine if this is a recursive call. If so, the recursive count is incremented, registers restored from the recursion save area, and the appropriate VDMEP recursion path taken. Prior to calling any module, except ABEND, recursion will be set up as follows:

> In register 14 will be placed the address of the recursion handling routine; a pointer to the appropriate error message to be output will be set up, and all registers saved in a special recursion save area. If recursion occurs, all registers are restored from the recursion save area, and a branch made to the recursion routine. This routine sets the 'ABEND required' switch, picks up the pointer to the specified message and goes to the exit subroutine.

If the Task ID = 2 (BULKIO) the BULKIO ABEND Recovery routine is called to attempt recovery.

For a Task ID = 1, or pre-logon/post-logoff calls, ABEND is required, after appropriate processing (QK004 - QK0065 - QK8500).

DCBID and DCB Header checks are made and either a diagnostic or an information message is inserted in the main message list (QK0100 - QK0315).

For a system data set, exit is via QK8500. Otherwise, the special processing flag is cleared and exit is via the VDMEP entry point to ABEND, CZACP3. Normally, ABEND will return to VDMEP at CZCQK3.

For entry at CZCQK3: ABEND has completed processing the VDMEP request. VDMEP must now operate on the data set in error. Entry at CZCQK3 is from ABEND via an entry in the AIR Table.

Processing of unopened data sets is shown from (QK3040 - QK3700).

If this is the secondary copy of a duplexed data set, the primary copy is found and closed (QK3041 - QK3044).

If a new data set was being created, the DSCB slot and catalog entry are deleted (QK3055 - QK3100).

Search SDST is called to close the entry in the SDST, if the data set is not duplexed and there are no DCBs open for the data set by this task (QK3200 - QK3210).

If there are no other DCB headers, the RESTBL pointer is cleared (QK3220 - QK3570 - QK3700); but if other DCB headers exist, the RESTBL is locked if necessary, and any existing buffer, overflow and indexed sequential directory pages are freed. The RESTBL lock is released and the RESTBL disconnected if necessary. If there are no other users, the RESTBL pages are freed (QK3300 - QK3570).

Diagnostic messages are output to SYSLOG and if there are no open DCBs, ABEND is called (QK9100 - QK9510).

Processing of opened data sets is shown in Chart JC (QK3900 - QK4700).

VAM ABEND Interlock Release is called to release interlocks, and if no STOW is necessary or allowed, the data set is closed with either CLOSE or DUPCLOSE, and the message/exit phase of the routine is entered (QK3900 - QK4050 - QK4500 - QK4700 - QK9100).

If a STOW is necessary, a STOW-R is done on an old member (QK4050) and a STOW-N done on a new member - or on the old member if the STOW-R didn't take (QK4100 - QK4350).

If the task is conversational the user is prompted for the member name (QK4250 - QK4270).

If the user has had four PRMPTs or defaults or if the task is nonconversational, a unique member name is created by VDMEP and placed in a message to inform the user of the name. This 8 character member name will consist of 4 alphabetic characters and 4 numeric characters; 'AAAANNNN'. Initially 'NNNN' = 0000; if a return code from CZCOK (STOW) indicates that this member name already exists, 'NNNN' is incremented by 1 and the call to STOW issued (QK4300 - QK4350).

When the STOW-N is complete, the member name is written in SYSLOG for a nonconversational task (QK4450), and both conversational and nonconversational tasks CLOSE/DUPCLOSE the data set and enter the message/exit phase of the routine (QK4500 - QK9100).

There is a subroutine (also used In-line) which places the diagnostic/message and its inserts in the message list (QK9050 - QK9090).

Any messages for SYSOUT that can be written (SYSOUT, SYSMLF are open - PROMPT can be called) are passed via CZATJ (QK9100

- QK9150), and then written in SYSLOG via CZABQ (QK9200 - QK9250).

ABEND is called if required at CZACP1, returned to normally if a previous call to CZACP3 was made from VDMEP, or called at CZACP3 if no previous call occurred (QK9500 - QK9600).

For entry at CZCQK2: CZCQK2 is entered from ABEND, when ABEND receives a recursive call while processing a VDMEP request. The 'ABEND Required' switch is set and the message/exit phase entered.

VAM INTERFACES

VAM effects the input/output of data by interfacing with the paging supervisor. External storage of a VAM data set is limited to direct access devices, whose records are in the page (4096 bytes) format used with that device.

VAM data sets are organized by relative page number. Each page of a data set is assigned a page number which is relative to the beginning of the data set.

These relative page numbers are translated to an input/output device address through use of the relative external storage correspondence table (RESTBL). The content of the RESTBL is created from data set extents obtained from data set control blocks (DSCBs) and maintained within virtual storage by VAM routines. External device addresses supplied by the RESTBL are passed to the paging supervisor, as required, to build the external page tables. In part, these are pointers to external storage areas associated with the active pages of a VAM data set.

One or more pages are required for the RESTBL. If a partitioned organization data set is opened, the partitioned organization directory (POD) will reside in virtual storage. For an index sequential data set, directory page(s), plus possibly an overflow page, will exist in virtual storage in addition to a one-page data buffer. With VSAM, a buffer in virtual storage is provided which is large enough to contain the largest record in the user's data set, with a maximum size of one segment (256 pages).

VAM is designed to minimize the number of virtual storage pages associated with an open data set. Only those data set pages currently being operated on by the user's program are addressable as virtual storage.

The virtual access methods routines interface with other routines in TSS/360 including some in the command system, catalog services, and the resident supervisor. These external routines are referenced in the module descriptions of the access methods routines.

MODULE ATTRIBUTES

All modules of the VAM have the following attributes:

READ-ONLY	The storage protection key is set to prevent the user from performing a store operation on any part of the CSECT.
REENTERABLE	More than one task may concurrently execute the code embodied in the CSECT.
PRIVILEGED	The CSECT will be protected against any reference by nonprivileged routines: the CSECT, however, may reference any part of VM.
PUBLIC	Available to all tasks.
FIXED	The size assigned will not vary while in execution.
SYSTEM	User reference to the module is prohibited, except through SYS symbols. SYS symbols are used to label entry points to nonprivileged system routines to which the user may transfer control by a standard CALL linkage.

LINKAGE CONVENTIONS

Seven modules of VAM are considered "fence sitters." That is, they may be called via type-1 linkage by either a privileged or a nonprivileged routine. Calls from those modules to privileged modules will be type-2 if it is necessary to "cross the fence."

The "fence sitters" are:

GET	CZCOR	VSAM routines
PUT	CZCOS	
SETL	CZCOT	
PUTX	CZCOU	
GET	CZCPA	VISAM routines
PUT	CZCPB	
SETL	CZCPC	

The routines referenced by the above modules may sometimes be called by type-2 linkage. In order to effect type-2 link-

age, V-cons and R-cons for the following modules are stored in the enter table (CHBET1). The code to access those modules is also given.

Name	Entry Point	ENTER Code
MOVEPAGE	CZCOC1	X'4C'
INSPAGE	CZCOD1	X'48'
DELPAGE	CZCOD2	X'49'
PUT	CZCOS3	X'3E'
FLUSHBUF	CZCOV1	X'4D'
GETPAGE	CZCPI1	X'47'
	CZCPI2	X'4E'
	CZCPI3	X'4F'
ADE	CZCPL1	X'46'

Other VAM modules may be called by either privileged or nonprivileged routines but are always executed in the privileged state. Those modules are also listed in the enter table:

Name	Entry Point	ENTER Code
FIND	CZCOJ1	'44'
STOW	CZCOK1	'45'
ESETL	CZCPD1	'41'
READ/WRITE	CZCPE1	'40'
RELEX	CZCPG1	'42'
DELREC	CZCPH1	'43'

CONTROL BLOCKS

Control block descriptions in this PLM provide the following information to assist in the understanding of the Virtual Access Method.

- SYMBOL - The assembly mnemonic as it appears in the assembly listing, DSECT listing, or module descriptions in this manual.
- DATA - A code to indicate the format of the information stored in a field. The possible values are listed in Table 24.

Table 24. Abbreviations Used in Control Block Descriptions

A	Address of an area-control block, subroutine, etc.
B	Relative byte position within an area.
C	EBCDIC data.
D	Relative doubleword within a control block.
L	Lock byte to be referenced by Test and Set (TS) instruction. See INTLK (CZCOH) module description for more information.
N	Number -- count, limit, size, etc.
R	R-con -- address of a PSECT.
V	V-con -- address of an entry point to a CSECT.
W	Relative word position within a control block.
X	Code defined in hexadecimal, or a group of fields.

- DESCRIPTION - A brief description of the contents and usage of a field. For code fields, a list of possible vaues is also given.

Control blocks and tables common to all VAM access methods are described in this section. Elements which are used only by a single VAM access method (such as VISAM) are presented with the discussion of the appropriate routine.

Interruption Storage Area (ISA) -- (CHAISA)

The interruption storage area is located at virtual storage addresses 0 through 4095. One copy exists for each task. It is used as a fixed communication region for interruption processing between the task and the task monitor, and between the task and the resident supervisor (Table 25).

Task Data Definition Table (TDT) -- (CHATDT)

The TDT specifies the data set name, and supplies information about the external storage of the data set. This control block is generated prior to OPEN time by either a DDEF command or a DDEF macro instruction. It is updated at open time if necessary (Table 26).

Table 25. Selected Fields of the Interrupt Storage Area

Symbol	Data	Description
ISAVMP	A	Virtual storage packing origin
ISANAS	N	Next available segment
ISATDT*	A	Task Data Definition Table origin
ISASPN	N	Shared Page Table number of the public segment
ISATDY	A	Dynamic Loader Task Dictionary
ISASDS*	A	Shared Data Set Table (SDST)
ISAVTH	X	Authority code
ISALCK*	L	Task Interruption Inhibit lock byte
ISACVP	X	Current VPSW
*Used directly by VAM.		

Table 26. Selected Fields of a JFCB

Symbol	Data	Description
TDTDDN	C	DDEF name (ddname)
TDTDS1, TDTDS2	C	Data Set name
TDTDSV	X	Data Set Organization X'04' VISAM X'05' VSAM X'06' VPAM
TDTDSR	C	Absolute generation number
TDTDSM	C	Member name
TDTOPN	N	Number of open DCBs
TDTVPY	X	Privilege flag; X'01' = privileged
TDTAQL	X	Access Qualifier X'00' unlimited X'01' read/write X'02' read only
TDTSHC	X	Sharing qualifier; X'01' = shared
TDTDEB	A	Pointer to RESTBL
TDTDCB	X	First 32 bytes of DCB
TDTVF1	X	Volume flag
TDTID1	C	Volume serial number
TDTDSC	A	Pointer to format E DSCB
TDTDUP	A	Pointer to secondary JFCB of a duplexed set
TDTSD1	A	Symbolic Device Allocation table (SDAT pointer)
TDTID2	X	2nd and 3rd volumes; same format as for TDTVF1, TDTID1, and TDTSDI, above
TDTAPN	X	Chain flag
TDTAPP	A	Chain to JFCB appendage

Data Control Block (DCB - CHADCB): (Figure 16 and Tables 27, 28, 34, and 35).

The data control block, generated by a DCB macro instruction, is used to maintain information necessary for access method routines to process a data set. It contains data set organization, record format, current page number, last operation, retrieval address of the current record, and V-cons and R-cons of the access method routines (macro transfer list) to process the data set or member.

A DCB is generated at assembly time by the DCB macro instruction. Subsequently, both the programmer and the system may enter information into the data control block fields. The process of filling in the DCB is completed at execution time.

Sources of information for DCB fields are, the DCB macro instruction in the source program, the DDEF command or macro instruction in the job stream (or DDEF macro instruction executed by the user program), and the DSCB.

These sources are used in that order and only fields not yet specified can be filled from each source. For example, if a field is specified in both the DDEF command or macro instruction and the DSCB, only information supplied by the DDEF command or macro instruction is used for the DCB; the corresponding field in the DSCB is ignored. The programmer can write routines that modify any data control block field.

The DCB for VAM is composed of five parts. The first part of the DCB is common to all access methods. The four remaining parts pertain to VAM only. Their relationship is shown in Figure 16.

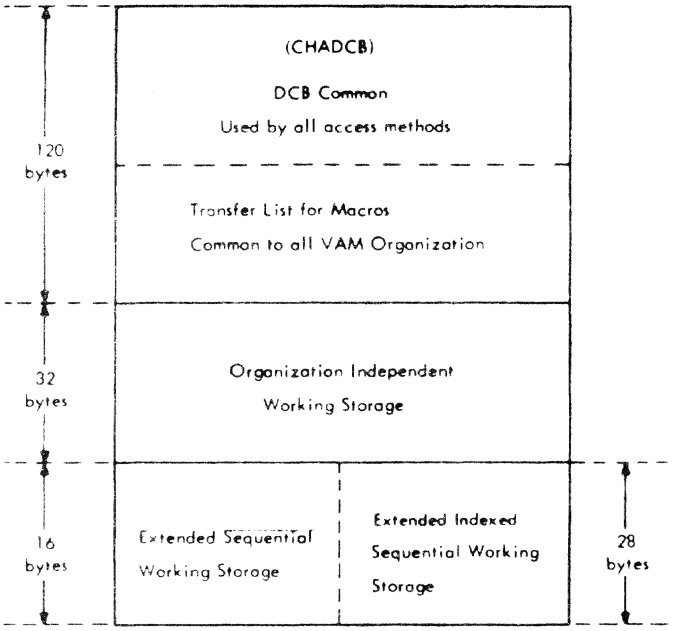


Figure 16. DCB Format for VAM

Relative External Storage Correspondence Table (RESTBL)

A control block used exclusively by VAM is the relative external storage correspondence table (RESTBL). A RESTBL is associated with each open data set using the virtual access method. It contains a list of external pages assigned to the data set. The RESTBL is used to convert page numbers relative to the data set, to external storage addresses. It also maintains control over data set page sharing. The RESTBL is located in an area of virtual storage protected from the user. The area of virtual storage that contains the RESTBL has a read-only protection key assigned. In the case of a shareable data set, the RESTBL pages are shared by user tasks, VAM sharing rules are discussed in the section on sharing.

A RESTBL is composed of four subsections whose functions are described below. The overall relationship is shown in Figure 17.

The RESTBL header contains control information for using the RESTBL.

The second subsection, (Figure 18), consists of a series of external page entries (EPE), with the control block identification CHAEPE.

One entry exists for each page of the data set, and contains the relative volume number and relative external page number, plus the page status, defined as in use, not in use, or assigned but not yet written.

Table 27. Selected Fields of the DCB Common

Symbol	Data	Description
DCBDSO	X	Data Set Organization Macro Code Param Access Routines DCBDV1 X'71' VIS VISAM DCBDV2 X'72' VS VSAM DCBDV3 X'73' VIP VISAM & VPAM DCBDV4 X'74' VSP VSAM & VPAM DCBDV5 X'75' VP VPAM & * *As determined by member organization
DCBDDN	C	DDEF name (ddname)
DCBSYV, DCBSYR	V,R	Synchronous Error exit address (SYNAD)
DCBEOV, DCBEOR	V,R	End of Data exit address (EODAD)
DCBRFC DCBRCF DCBRCV DCBRCU	X	Record format X'80' Fixed X'40' Variable X'C0' Undefined
DCBLRE	N	Record length
DCBKEY	N	Key length
DCBRKP	N	Relative key position
DCBLPA	N	Retrieval address
DCBEX1, DCBEX2	X	SYNAD codes
DCBOPI	X	OPEN options
DCBID	C	DCB identifier: C'*%*%'
DCBDEB	A	Pointer to RESTBL
DCBLEN	N	Length of this DCB (in doublewords)
DCBGTV, DCBGTR	V,R	GET routine*
DCBPTV, DCBPTR	V,R	PUT routine*
DCBPXV, DCBPXR	V,R	PUTX routine*
DCBSLV, DCBSLR	V,R	SETX routine*

*R-cons -- Pointer to save areas on a dynamically allocated page

Table 28. Description of the Fields Comprising the VAM Organization Independent Working Storage

Symbol	Data	Description
DCBVMA	A	VMA of next record in buffer
DCBCPB	N	Current page and byte; defined as follows:
DCBDPN	N	Current data page number
DCBCBP	N	Byte position relative to current page
DCBN	N	First page in request
DCBM	N	Number of requested pages
DCBOP	X	VAM General Services Operation:
DCBOP0		X'8000' Input, set Read interlock
DCBOP1		X'4000' Loader request
DCBOP2		X'2000' Input, set Write interlock
DCBOP3		X'1000' Output
DCBOP4		X'0800' Insert
DCBOP5		X'0400' Delete
DCBOP6		X'0200' Release Read interlock
DCBOP7		X'0100' Release Write interlock
DCBOP8		X'0080' Replace blank pages on an insert
DCBHV	N	Hash value of member name (6 bits)
DCBNI	N	First page in request, relative to data set - computed by VPAM routine GETNUMBR (CZCOO)
DCBSHC	C	Type of search request; see SEARCH (CZCOL)
		C'A' Alias name
		C'E' Either alias or member name
		C'M' Member name
DCBHD	A	DCB header in RESTBL

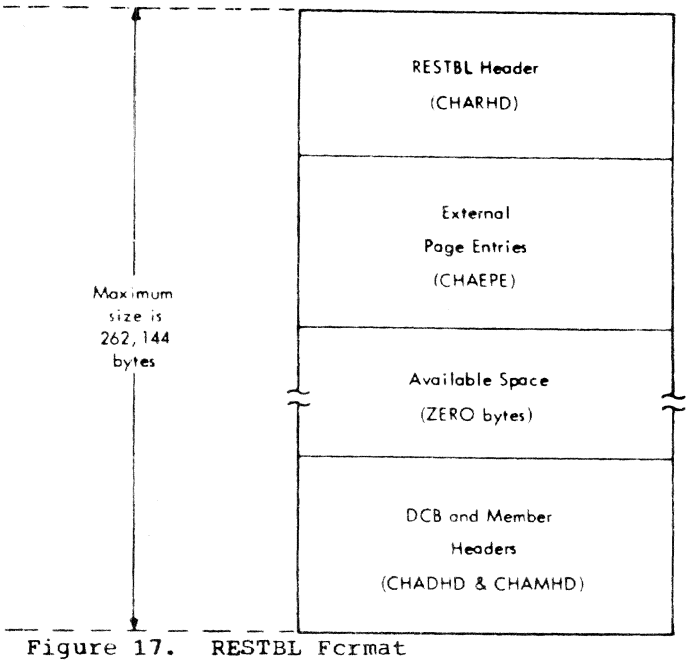


Figure 17. RESTBL Format

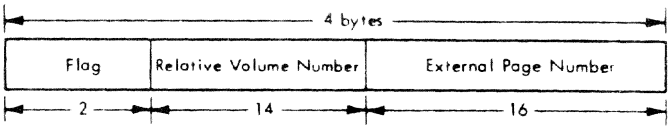


Figure 18. RESTBL External Page Entry - CHAEPE

For shared data sets, a 4-byte interlock word in each EPE controls sharing of the page. The interlock control word, and method of updating it, is discussed in detail in the sections on VAM sharing.

The third and fourth subsections of the RESTBL,, designated as DCB and member headers (CHADHD and CHAMHD), are constructed at open time. Address pointers in the DCB header identify associated control blocks: DCB, JFCB, POD, RESTBL header. The DCB header is linked to the RESTBL header or the member header (if partitioned organization) by DHDLNK.

The DCB header link, and compatible field design of the RESTBL and member headers, make it possible to process members of data sets identically.

A RESTBL is created for a data set by the OPENVAM (CZCOA) routine, using information extracted from the DSCBs of the volumes where the data set resides. Each volume contains all data set control blocks (DSCBs) for the data sets contained on that volume. At open time, the external page entries (EPE) are built from extents found in the DSCB. As the data set is generated, additional extents are obtained dynamically

by REQPAGE (CZCOE). Once a shared data set is open, no new RESTBL is generated by any subsequent OPEN, that is, only one RESTBL ever exists at a time for a shared data set. However, for each DCB opened for a data set, one DCB header will exist in the RESTBL. If this is a shared, nonpartitioned data set, each user's DCB will have a DCB header. Each of these headers, in turn, is linked to the RESTBL header.

For a partitioned data set, the DCB headers of each open DCB will be linked to member headers.

The basic purpose of the RESTBL header or member header is to document three items of importance:

- ORGANIZATION - Sequential or Index Sequential
- RECORD FORMAT - Fixed, Variable or Undefined
- CONTENT - Starting page position in the RESTBL, number of data, overflow, or directory pages

The RESTBL header also accounts for pages assigned to the data set, but not yet in use, as well as available virtual storage in the RESTBL.

Closing a DCB causes its DCB header to be deleted from the RESTBL. In a partitioned data set, this also causes the member header to be deleted provided no other DCB headers exist for that member. The RESTBL header and external page entries will remain in virtual storage until the last DCB is closed. At that time, the CLOSEVAM (CZCOB) routine will return the contents of the RESTBL to the DSCBs associated with the data set.

Descriptions of fields for the RESTBL header (CHARHD) and for the DCB header (CHADHD) are provided in Table 29 and Table 30, respectively.

Shared Data Set Table (SDST)

The SDST, whose address is given in the interruption storage area (CHAISA) of each task, consists of a header (CHASDS) and a series of data set (CHASDE) and member (CHASDM) entries. The SDST format is illustrated in Figure 19. Tables 31, 32, and 33 provide field descriptions of the SDST header, a member entry and a data set entry. The data set entries are linked by forward and backward chain pointers, with the pointer to the first data set entry in the SDST header. Member entries are organized into 64 hash chains. The hash chain to which a member is linked is generated from the member name. A table of 64 words, part of the SDST header, gives the address of the first member within each hash chain. In addition to the data set and member chains, two chains of deleted entries are maintained in order to recover space for building data set or member entries. Data set and member entries are linked to the appropriate deleted chain when their user count reaches zero.

The information in the data set and member entries is used to control access to a data set, and also to provide a common location to store the information (shared page table number and RESTBL address) necessary for multiple tasks to obtain access to an existing control block in shared virtual storage. This control block is updated by the VAM general services routine Search SDST (CZCQE) which has the following capabilities:

- Search the SDST for a specified data set entry and/or member entry.
- Modify data set or member entries by incrementing or decrementing the user count. This capability also provides for creating or deleting such entries.

The linkages between a user's DCB and the RESTBL, POD, and JFCB for a member of a partitioned data set, are shown in Figure 20. DHDLNK is shown as linked to either a member header or the RESTBL header, since this field, when the data set is nonpartitioned, will point to the RESTBL header. If this data set were shareable, the RESTBL and POD would be in shared virtual storage.

SDST MAINTENANCE

The following routine maintains the shared data set table (SDST).

Search SDST Routine (CZCQE)

The Search Shared Data Set Table routine is called by OPENVAM (CZCOA), CLOSEVAM (CZCOB), Find (CZCOJ), Stow (CZCOK), and the dynamic loader, to add, update or delete, data set or member entries in the shared data set table (SDST), and establish correspondence to shared virtual storage. (See Chart JD.)

Entry Point: CZCQE1 - Entered via type-1 linkage.

Input: Register 1 contains the address of a four-word parameter list:

Word 1 -- Address of JFCB.

Word 2 -- Address of DCB.

Table 29. Field Descriptions for the RESTBL Header -- (CHARHD)

Symbol	Data	Description
		The first 4 bytes of the RESTBL form a VAM interlock word which is updated by INTLK (CZCOH) and RLINTLK (CZCOI)
RHDINW	L	Write interlock
RHDINR	N	Read interlock
RHDINN	N	Count of read interlocks set
RHDINI	L	Update interlock
RHDNAP	W	Next available External Page Entry (EPE)
RHDNEP	N	Count of available EPE
RHDFEP	W	First EPE of data set
RHDDIR	N	Size (pages) of the Indexed Sequential Directory or Partitioned Organization Directory (POD)
RHDDAT	N	Data set size (pages)
RHDOVF	N	Number of overflow pages
RHDRPG	N	RESTBL size (pages)
RHDTHD	W	Last header built from available space. This field, when decremented by the size of the header to be built (DCB=48, member=32) gives the address where a new header may be built
RHDFLG	X	Data set organization: X'80' Shared X'40' Partitioned X'20' Index Sequential X'08' ISD Integrity X'04' POD Integrity X'02' DSCB Integrity X'01' Recatalog Flag

Symbol	Data	Description
RHDINI	L	Lock byte for the next 6 fields
RHDDCB	N	Number of DCBs that are OPEN
RHDODC	D	First DCB header
RHDADC	D	First deleted (available) DCB header space
RHDOMC	D	First member header
RHDAMC	D	First deleted (available) member header space
RHDPOD	A	POD
RHDTID	D	Task ID which set RESTBL interlock
RHDVTA	A	Address of the volume table
RHDSPT	X	Pointer to the format E DSCB
RHDCPO	N	Data set cumulative pageout count
RHDSAL	A	Secondary allocation (ESA)
RHDRFM	X	Record format
RHDKYL	X	Key length
RHDPAD	X	VI pad factor
RHDRKP	B	Relative key position
RHDRCL	N	Record length
RHDDSO	X	DSORG
RHDCRD	X	Change/Reference Date flags
RHDOPC	X	Option codes

Table 30. Field Descriptions for the DCB Header -- (CHADHD) (Part 1 of 2)

Symbol	Data	Description
DHDDCB	A	DCB associated with this header
DHDJFC	A	JFCB (TDT) for the data set
DHDTSK	N	Task identification
DHDRES	A	RESTBL address
DHDPOD	A	POD address
DHDLNK	A	Linkage to either the RESTBL header or to the member header if this is a partitioned data set and a member is active (FIND has been done)
DHDOPN	X	OPEN options - same as DCBOP1
DHDPRO	X	Protection class of the virtual storage in which the DCB resides X'03' Read only X'01' Read/Write X'07' Private privileged
DHDINT	X	Interlock summary. This field is updated by INTLK (CZCOH), RLINTLK (CZC01), and VAMABIR (CZCQQ) The following reflect interlocks set in the indicated control blocks: X'0001' Data Set Entry in SDST X'0002' RESTBL header X'0004' Member descriptor in POD X'0008' POD X'0010' EPF that are checked out to this DCB X'0020' Member header in RESTBL X'0040' SDST control entry X'0080' RESTBL header partial interlock set (RHDINI) X'0100' SPT number in SDST Data Set Entry locked Some of the above interlocks indicate either a read or a write interlock. The next 5 mask values indicate which of the control blocks are write interlocked.

Table 30. Field Descriptions for the DCB Header -- (CHADHD) (Part 2 of 2)

Symbol	Data	Description
		X'0200' SDST data set entry X'0400' RESTBL header X'0800' Member descriptor in POD X'1000' POD X'2000' External Page Entries
		The following 4 fields are used with VSAM organization:
DHDFBP	A	Buffer address
DHDNBP	N	Buffer size (pages)
DHDFDP	N	First data page checked out to this DCB
DHDPCO	N	Number of pages checked out to this DCB
DHDCOP	A	Overflow buffer
DHDISD	A	VISAM directory location
DHDCDP	N	Current data page
DHDNOP	N	Current overflow page
DHDMRL	N	Maximum record length
DHDNDH, DHDPDH	W	Forward and reverse DCB header pointers
DHDDUP	A	Address of duplex copy of the RESTBL
DHDDXP	A	Current External Page Address
DHDOXP	A	Overflow External Page Address

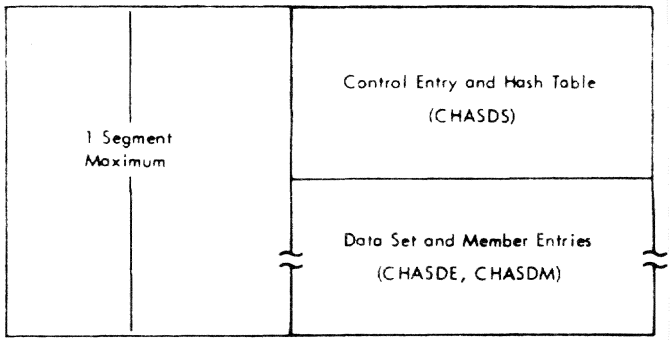


Figure 19. Shared Data Set Table (SDST) Format

Table 31. Field Description of the SDST Header -- (CHASDS)

Symbol	Data	Description
SDSINT	L	SDST interlock byte. If set, access is not permitted
SDSLPN	N	Last page number assigned to the SDST
SDSSPT	N	Last assigned Shared Page Table (SPT) number
SDSAVA	A	Next byte of available space in the SDST
SDSDE	A	First deleted Data Set Entry
SDSDME	A	First deleted member entry
SDSSDE	A	First Data Set Entry
SDSHAS	A	First member entry for each of the possible 64 hash values of member names
SDSPLK	L	VMA lock on SDSPSN
SDSPSN	N	Public segment numbers

Table 32. Field Description of a Member Entry -- (CHASDM)

Symbol	Data	Description
SDMCHN	A	Next member entry in this hash alias chain
SDMNUR	N	Number of users in this member
SDMSPT	N	Shared Page Table (SPT) number assigned to this member
SDMNSP	N	Number of shared pages
SDMFSP	N	Number of the first shared page
SDMLSD	A	Data Set Entry corresponding to this member
SDMNAM	C	Member name

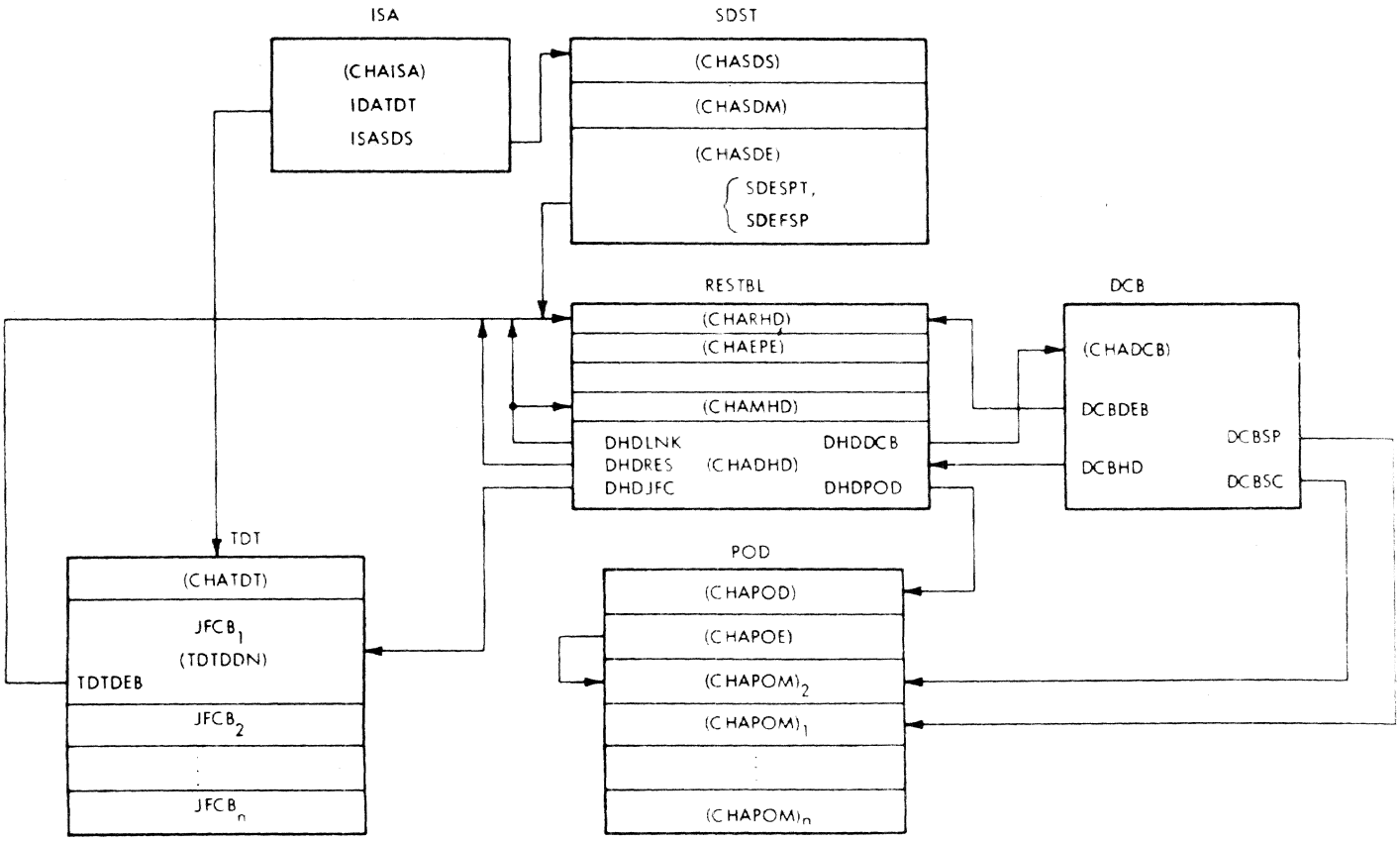


Figure 20. Linkage Relationships Among Control Blocks Used with VAM

Table 33. Field Description of a Data Set Entry -- (CHASDE)

Symbol	Data	Description
SDECHN	A	Next data set entry The next 4 bytes form a VAM interlock word which is updated by INTLK (CZCOH), and RLINTLK (CZCOI)
SDEINW	L	Write interlock
SDEINR	L	Read interlock
SDEINN	N	Count of read interlocks set
SDEINI	L	Update interlock
SDENUR	N	Number of users
SDESPT	N	SPT number
SDENSP	N	Number of shared pages
SDEFSP	N	Number of the first shared page entry
SDENAM	C	Data Set name

Word 3 -- Address of 8-byte member name, zero if no member.

Word 4 -- Address of 1-byte type code:

C - close

O - open

U - user count only

Modules Called:
Interlock (CZCOH1) -- Set write (W) interlock on a shared data set entry (SDSE).

Release Interlock (CZCOI1) -- Release write (W) interlock on an SDSE.

ABEND (CZACP) -- Abnormal termination of task.

Exits:
Normal -- For Open option:

'00' Entry existed.

'08' New entry built.

'10' Data set does not exist.

For Close option:

'00' Entry closed, users still exist.

'04' Entry closed, no more users.

'10' Data set does not exist.

Error -- ABEND is called when one of the following conditions exist:

- No pointer to the DCB.
- SDST or shared page table already locked by this task.
- The data set is non-VAM.
- Current pointer is higher than the "last page" pointer.
- There is insufficient space for the member entry or data set entry.
- An attempt is made to close an unknown member or an unknown data set.
- An attempt is made to access a non-shared data set by two different tasks.

Operation: Overall operation of this module consisits of entry linkage, data set name search, member name search, analysis of user requirements, updating of data set entries and member entries, and either a normal return, an error return, or an ABEND.

After completion of linkage and loading base registers from the parameter list, this module checks that the DCB code is present in the field DCBID. If unequal, ABEND is invoked.

By use of test and set (TS) and time slice end (TSEND), execution of this task will be delayed until the SDST interlock byte is found in the reset state. At that time, this task has exclusive control of the SDST, and a search through the chain of data set entries is made beginning with the SDSE specified in SDSDSE, by comparing the 44 byte data set name in the JFCB with the data set name in each SDSE in the chain.

If the end of the SDSE chain is reached before a matching data set name is found, or if no chain exists, an SDSE for that data set name does not exist--subsequent action depends on the option requested by the calling routine in the parameter list. When a matching name is found, this task will delay execution until the shared page table number (SDESPT) is not equal to X'FFFF' (SPT lock condition). If the lock condition is present, and this module was not called by ABEND for release of interlocks, the portion of the SDSE search for the specified data set name which follows interlocking the SDST, will be repeated. Then if a member name was specified in the

list of input parameters, a search for that member name entry will be performed.

The 8 byte member name is hashed to 6 bits using the logical operation "exclusive or". That hash value is used to pick one of the 64 member entry chains, and the selected chain is searched for the specified member name. Each member name entry in the selected chain is compared against the member name specified in the parameter list. If equal, the data set name in the SDSE addressed by the member entry is compared. If both are equal, the proper member entry has been found and will be processed by this routine. If no member entry chain exists or no matching member name is found or matching member names do not point back to the SDSE previously found, then no member entry exists. Continued processing depends on options specified in the parameter list.

The following describes processing following data set and member name searches.

If the SDSE was not present, and a member name was specified, an error return is made. If a member name was not specified, and the option was CLOSE, an error return is made. Otherwise, a new SDSE is built, whether from the chain of deleted SDSEs or from available storage, and linked to the active SDSE chain with the user count set to one. Insufficient storage will cause ABEND. For VSAM or VISAM organization, Interlock (CZCOH1) is called to impose a write interlock on the SDSE if the OPEN option is non-Input. A return is then made to the user with a normal completion code and the address of the found SDSE in general register 1.

If the SDSE was present and a member name was not specified, the organization and operation will be checked prior to updating the SDSE. If this is not a VAM organization, ABEND occurs. If this is either a VSAM or VISAM data set (but not VPAM) to be opened, for OUTPUT, OUTIN, or UPDATE, the presence of an SDSE indicates that this data set has at least one user and is therefore not available to this task. If a VSAM or VISAM data set is to be opened for INPUT and a write interlock exists, the data set is not available. For either of these two cases, the interlock on the SDST will be released, and TSEND will be used to delay processing of this task until the other tasks have ceased to process that data set. Each time the task is reinitialized, the procedure to search for the SDSE will be started at the point (described above) where an interlock on the SDST is established by use of a TS.

Where a VSAM or a VISAM data set is to be opened for input and a write interlock does not presently exist, Interlock is called to impose a write interlock on the SDSE if the open option is other than INPUT. It is not necessary to impose a read interlock, since the presence of an SDST entry without a write interlock implies a read interlock condition. The last operations to be performed are, to increment the user count in the SDSE, place the normal return code in register 15, the address of the SDSE in register 1 and return to the calling routine. This is done for VSAM or VISAM opened for input, and for any VPAM open.

When the specified operation is CLOSE and no SDSE exists, the interlock on the SDST is released, and an error return is made to the calling program.

When the specified operation is CLOSE and the SDSE was found and a member name was specified, the count of users will be reduced. If this count reached zero, the member entry will be deleted from the member chain and its space appended to the deleted member chain. A return will be made to the user with a normal completion code.

When the operation is CLOSE, the SDSE was found and no member name was specified; if the data set organization is VSAM or VISAM, Release Interlock is called to release the interlock that had been set on the data set-R read or write according to the OPEN option in the DCB. Following that, the user count in the SDSE is reduced by 1. If the count reaches zero, that SDSE is deleted from the chain of active SDSEs, and its space appended to the chain of deleted SDSE. A normal return is then made to the user.

All direct access devices (2311 or 2314), on which VAM data sets reside, are initialized and maintained in a standard format. All space on these volumes, with the exception of cylinder 0, track 0, and possibly cylinder 0, track 1, is treated in units of 4096 bytes (one page). Associated with each volume is a page assignment table (PAT) which contains entries for all of the pages on the volume. The length of the PAT depends on the type of device and currently occupies one page on the 2311 volume, and two pages on the 2314 volume. This table is pointed to by a field in the volume label.

Each entry in the PAT is 1 byte long. If the first bit of the entry is zero, the page represented by the entry is either available for assignment to a data set or is already assigned. The remaining 7 bits provide a binary count of the number of data sets sharing the page. When this count is zero, the page is available for assignment.

If the first bit of the entry is 1, the page it represents is either a DSCB page or an error page. If the second bit of such an entry is 0, the entry represents a DSCB page, that is, the page contains data set control blocks (DSCBs). Each DSCB page can contain up to 16 DSCBs and is, accordingly, divided into slots. When only 4 of these slots remain available on a page, one of two flag bits is set to one. This indicates that the page is only to be used for DSCBs which further describe data sets already included in that DSCB page. The second of these flags is set to 1 when all 16 slots on the page are in use. The function and format of the DSCBs will be discussed further below.

The third type of entry which can be found in the PAT is the error page entry. This entry represents a page which is unsuitable for use due to surface errors discovered by the RESTORE utility and is indicated by a 1 in bits 0 and 1. Following the page entries in the PAT, there are several unused bytes of space. The last 97 words of this space are used for relocation entries. The last word in the table is a relocation control entry, which maintains a half-word count of the number of relocation entries which precede it, and a half word (X'FFFF') if there are any entries. The other 96 words may be occupied by relocation entries. These entries contain the relative page number of the error page, and the relative page number of the relocation page.

THE DATA SET CONTROL BLOCK (DSCB)

Associated with each data set is one or more DSCBs. The first of these control blocks is called a format-E DSCB; if additional space is required to describe the data set, format-F DSCBs are built and chained to the format-E DSCB. A data set and the DSCBs which describe it may be on more than one volume. Because of this, the chaining procedure used gives the relative volume number and the relative page number on that volume where the next DSCB is located. Also, since each DSCB page contains up to 16 DSCBs, the slot number of the DSCB is included. Each format-E DSCB contains the data set name and properties and possibly the external page entries which give the location of the data set pages on external storage. These entries contain the relative volume and relative page number on the volume where the data set page is located.

The concept of relative volume numbers, introduced above, arises from the possibility of a data set occupying more than one volume. For public data sets, this relative volume number provides an indexing factor into the public volume table. This table consists of a 16 byte header, which contains a count of the maximum number of public volumes allowed, and the count of public volumes actually in use. Following the header is a series of 16 byte entries representing each volume assigned to public storage. The entry contains the volume ID, the device code, and the symbolic device address of the volume. This information, in conjunction with the pathfinding tables and the pathfinding routine of the resident supervisor, make it possible to locate all volumes which contain a given data set.

The public volume table is a separately assembled CSECT which is a part of initial virtual storage and is initialized by Startup. In order to maintain similarity in the processing of public and private data sets, a table, called the private volume table, is built for private volumes. The address of the volume table is placed in the RESTBL header by OPENVAM.

The content of the DSCBs for a data set also vary according to whether the data set is public or private. Within the field of data set properties is a count of the numb-

er of volumes which the data set occupies. For public data sets this count is zero since the list of volumes is contained in the public volume table. In this case, the data set name and properties field is followed immediately by a list of external page entries. Each of these entries consists of a relative volume number which provides an index into the public volume table and a relative external page number, which provides an index into the page assignment table on that volume. For private data sets residing on one volume, this field is also zero since the volume ID is contained in the data set descriptor. For multivolume private data sets, the data set name and properties field is followed by a list of the volume IDs on which the data set resides. This list is used by MOUNTVOL when it builds the private volume table. The format-E DSCB may contain up to 25 volume IDs each 6 bytes long. Following the list of volume IDs, is the list of external page entries which are the same as for public data sets.

BUILDING AND MAINTAINING A DATA SET

When a user is building or updating a data set, he conceives of it as being a group of contiguous pages of records. Actually, because of the virtual storage concept, the pages of any data set may be physically located in several external areas known as extents. In order to allow the user to continue to think of the data set in terms of contiguous pages, it is necessary to construct a relative page/external page correspondence table (RESTBL). This table is built by OPENVAM when a data set is first opened for use. The RESTBL consists of a header which contains information such as the number of pages available for assignment to the task, the number of pages currently in use by the task, and the relative location of the first unused page which can be assigned to the task. The balance of the RESTBL consists of a series of external page entries which are identical in format to the external page entries in the DSCB described above. This parallel construction of external page entries enables the system to assign and delete pages in a data set, and to update the availability of the pages on the volume, without the use of conversion routines.

The external pages assigned to a task are also placed in the user's external page table or shared page table so that a reference to a virtual storage address is translated to the correct main storage address during execution.

During execution, a data set may be dynamically increased or diminished in size or deleted completely. When this occurs, the affected pages must be added to or deleted from the data set and the task's RESTBL. For this purpose, six service routines are provided with the virtual access method:

Insert/Delete Page (CZCOD) -- Effects the addition or deletion of pages in a data set and performs error checks to determine the validity of the operation.

Insert (CZCOF) -- Adds external page entries representing the new pages to the task's RESTBL.

Request Page (CZCOE) -- Provides Insert with a list of available pages for insertion into the RESTBL. It also marks the pages in use and unavailable for assignment.

Expand RESTBL (CZCQI) -- Increases the size of the RESTBL for nonshared data sets when the addition of new page entries causes an overflow condition.

Reclaim (CZCOG) -- Deletes page entries from a task's RESTBL and adds them to the available list for future assignment or release at the time the data set is closed.

DELVAM (CZCFT) -- Deletes a virtual organization data set by deleting its catalog entry and freeing the external pages and DSCB slots which it occupies.

Insert/Delete Page Routine (CZCOD)

The Insert/Delete Page routine is called by FLUSHBUF (CZCOV), GETPAGE (CZCPI), and Add Directory Entry (CZCPL) to check and perform insertion or deletion of pages within a data set. The validity of the request is checked, based on the specific request and on parameters in the DCB and RESTBL. (See Chart KA.)

Entry Points:

CZCOD1 -- Insert pages, entered via type-1 or type-2 linkage.

CZCOD2 -- Delete pages, entered via type-1 or type-2 linkage.

Input: Register 1 contains the address of the DCB. Two fields which must be set in the DCB are:

DCBN -- Page number, relative to the data set or member, at which the operation is to take place.

DCBM -- Number of pages to be inserted or deleted.

Modules Called:

Insert (CZCOF1) -- Insert additional pages at a specified position within a data set and move all other active pages upward.

Reclaim (CZCOG1) -- Delete specified pages.

Interlock (CZCOH1) -- Set a write interlock in the RESTBL external page entry.

Release Interlock (CZCOI1) -- Release write information on RESTBL.

GETNUMBR (CZCOO1) -- Validate and perform insertion or deletion on a partitioned data set.

TSEND (CEAH19) -- Force end of time slice for this task, to wait for interlocks in shared pages to be released by the task that had set them.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

Exits:

Normal -- Return to the calling routine with one of the following return codes:

'00' Normal.

'04' No storage space available.

'08' Storage ration exceeded.

'0C' No secondary storage allocation specified.

'10' Shared data set RESTBL cannot be expanded.

'14' Maximum data set/or member size exceeded.

'18' Insertion beyond end of data set.

'1C' Deletion beyond end of data set.

Error -- VDMEP is called and the function (not the task) terminated if an invalid return code is received from Reclaim, Insert, or GETNUMBR.

Operation: Calls to CZCOD are for the logical insertion or deletion of data pages within a data set. If the routine is entered at the primary entry point (CZCOD1), the Insert flag in the DCB operation field (DCBOP) is set to insert (DCBOP4). If entry is made at the secondary entry point (CZCOD2) in the case of a deletion, no indication is set at this time. The code from these two types of entries converges to perform module initialization.

Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the calling program's save area. CZCOD CSECT and PSECT, DCB, DCB header and RESTBL.

Insertions or deletions may not be done on data sets which are opened for input only. If the DCB open option indicates input, ABEND is called immediately. The RESTBL is interlocked for shared data sets.

If the data set is partitioned, GETNUMBER (CZCOO) must be called. The requested operation is tested. If the Insert flag is not set as described in the entry procedure, the Delete flag (DCBOP5) is set and GETNUMBR called. Upon a successful return from GETNUMBR, control is returned to the user by the RETURN macro, since GETNUMBR has already accomplished the desired insertion or deletion.

For nonpartitioned data sets, CZCOD performs a great deal of the consistency and validity check required for the insertion or deletion.

The extent of a deletion, that is, the first page plus the number of pages being deleted, must be contained within the data set. If any of the pages to be deleted falls outside of the range of the data set, ABEND is called.

Reclaim (CZCOG) is called to accomplish the deletion. The return code from Reclaim is tested for errors. If general errors exist or the deletion was requested on pages of a shared data set that were interlocked, an appropriate return code is set and control returned to the caller. Upon successful deletion, control is returned to the caller by the RETURN macro. If the data set is shared, the RESTBL interlock is released before returning.

An insertion must be made within the range of the data set or contiguous to the last page. If the insertion is requested outside or not adjacent to the data set, an appropriate return code is set and control returned to the caller. The actual insertion of the data page is performed by Insert (CZCOF). The return code from Insert is tested for errors. If they exist, VDMEP is called. Control is returned to the caller by the RETURN macro. For shared data sets the RESTBL interlock is released before returning.

Insert Routine (CZCOF)

Insert is called by Insert Page (CZCOD), Extend POD (CZCOM), and GETNUMBR (CZCOO) to insert additional "in use" pages in the

RESTBL of the data set involved in the request. (See Chart KB.)

Attributes: Read-only, reentrant, privileged.

Entry Point: CZCOF1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB. Fields used in the DCB are:

DCBN -- First page of data set involved in request (nonpartitioned).

DCBNI -- First page of data set involved in request (partitioned).

DCBM -- Number of pages involved in request.

DCBOP -- Type of operation requested.

Modules Called:
Request Page (CZCOE1) -- Flags additional external page entries "in use" in the available page chain in the RESTBL. Request Page will also attempt to increase the number of pages available to the task, by calling FINDEXPG, when a sufficient number of pages is not available to fill the request.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

Exits:
Normal -- Return to the calling routine.

Error -- A valid error code returned from REQPAGE is passed back to the calling routine in register 15. An invalid error code returned from REQPAGE results in VDMEP being called.

Operation: When Insert is first entered, registers are saved and base registers established for the CSECT, PSECT, DCB, DCB header, and RESTBL header. Insert then retrieves the number of the first page to be inserted and the total number of pages to be inserted from the DCB. Next, the virtual storage addresses of the next available page and of the location of the insertion are computed, and a call is made to Request Page (CZCOE). Request Page will flag the requested number of pages "in use" by setting the flag in the external page entries in the RESTBL. If a sufficient number of pages are not available to fill the request, an attempt is made to add pages to the available list from external storage. If Request Page is unable to fill the request, an error return is made to the calling routine by the RETURN macro instruction.

Insert may be called to insert pages between existing pages of the data set or to simply add pages to the end of the data set. In the latter case, Request Page will have added the new pages at the end of the list of "in use" pages and no additional work is required of Insert. If pages are being added between existing pages, it is necessary to create a gap in the list of "in use" pages in the RESTBL. For this purpose, a 64-word work area is provided in the PSECT. This area will hold 32 shared or 64 nonshared external page entries. The new entries are moved into this work area starting from the beginning of the available list. All current "in use" entries following the point of insertion are moved back in the RESTBL an equal number of words to create a gap. The new page entries are then moved from the work area to the gap thus created. This procedure is repeated until all new entries have been inserted.

In the case of duplexed data sets, this process is repeated for the secondary copy, after which the DSCB integrity bit is set and a normal return is made to the calling routine . The setting of the integrity bit will cause CLOSEVAM to call Write DSCB to update DSCBs on the volumes. In this manner, pages dynamically added to the data set are made unavailable for future allocation.

Note that this routine does not interlock the RESTBL for shared data sets. This is so because Insert assumes that the calling routine had done so before calling it. Also, Request Page is responsible for updating the counts of external pages in use, available pages, and the relative location of the first available page.

Expand RESTBL Routine (CZCQI)

Expand Relative External Storage Correspondence Table is called by OPENVAM (CZCOA), MOVEPAGE (CZCOC), Request Page (CZCOE), and Find (CZCOJ) to increase the size of the RESTBL by one page. The additional space appears between the external page entry words, and the DCB and member header control blocks. Note that a RESTBL which resides in shared virtual storage cannot be expanded. (See Chart KC.)

Attributes: Read-only, reenterable, privileged.

Entry Point: CZCQI1 -- Via type-1 linkage.

Input: Register 1 contains the address of the RESTBL to be expanded.

Modules Called: VMA (CZCG4) -- To increase the size of a specified virtual storage area.

Exits:
Normal -- Execution of the calling program is resumed by use of RETURN. No special return code is given.

Error -- None.

Operation: Initialization and general register storage is executed in conformance with linkage convention. Base registers are declared for the CSECT and PSECT, a save area, RESTBL, DCB header and JFCB.

Expand RESTBL calls Expand (CZCG4) by the CALL macro to expand the virtual storage of the RESTBL by 1 page. After the expansion, the RESTBL pointers, are updated in the JFCB if the RESTBL was relocated in the expansion. RESTBL pointers (VM addresses) within the DCB and member headers are updated if the RESTBL has been relocated.

The DCB and member headers are moved on to the new page up to the top header; this creates space between the external page entries and headers, for the insertion of new information. The relative pointers within the headers, and the DCB and member headers are updated by 4096 to reflect their new position.

When the new virtual memory space has been obtained, and all pointers and addresses which need to be adjusted have been updated, Expand RESTBL returns to the calling module by the RETURN macro.

Note that a RESTBL which resides in shared virtual storage cannot be expanded.

Request Page Routine (CZCOE)

Request External Pages is called by Insert (CZCOD) and OPENVAM (CZCOA) for the purpose of assigning available external pages to a data set. If sufficient pages are not available to fill the request, an attempt is made to increase the size of the data set by calling FINDEXPG (CZCEL). The newly assigned pages are added to the task's RESTBL or, in the case of a shared data set, to the shared RESTBL. If this addition results in an overflow of the RESTBL, it is expanded to accommodate the extra pages in the case of nonshared data sets. Since a shared RESTBL cannot be expanded, this condition results in a call to VDMEP (CZCQK1). (See Chart KD.)

Attributes: Read-only, reentrant, privileged.

Entry Point: CZCOE1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB. Fields used in the DCB are:

DCBM -- Number of pages requested.

DCBN -- Relative number of the first page affected.

Modules Called:
Expand RESTBL (CZCQI1) -- Increases the size of the RESTBL for a nonshared data set when the inclusion of the newly assigned pages causes overflow of the current RESTBL.

FINDEXPG (CZCEL1) -- Assigns additional external pages to the task from the storage allocated to the task.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

Exits:
Normal -- RETURN to the calling routine if successful or if the request is for zero pages with one of the following return codes in register 15:

'00' Normal.

'04' No external storage available.

'08' Storage ration exceeded.

'0C' No secondary storage allocation.

'10' Shared data set RESTBL cannot be expanded.

Error -- VDMEP is called if an invalid return code is received from FINDEXPG (CZCEL1).

Operation: Initialization and general register storage is performed in conformance with standard linkage conventions and base registers are established for the CSECT, PSECT, DCB, DCB header, and the RESTBL. The number of pages involved in the request (DCBM) is checked and, if the request is for no pages, a normal return is made to the user.

The function of Request Page is to assign external pages to a task's data set from external storage which has been previously allocated. These pages may be assigned from either the primary or the secondary allocation if it ie be assigned from either the primary or the secondary alocation if it exists. Additionally, the page entries in the RESTBL are flagged "in use" and any pages which have been assigned are added to the RESTBL.

These operations differ in two ways for shared or nonshared data sets. In the case of shared data sets, two words of RESTBL space are required for each page entry; the additional word is occupied by the page

interlock required for shared data sets. The other difference is that the RESTBL cannot be expanded for shared data sets.

Following the initialization process, the data set is examined to determine if it is shared. With the exception of the above noted differences, the processing is the same; the exceptions will also be mentioned in the following description. In each case the RESTBL is tested (RHDNEP) to determine if sufficient pages are already assigned. If they are, the pages are flagged "in use" in the RESTBL external page entries, and the count of available pages (RHDNEP) and the location of the next available page (RHDNAP) are updated in the RESTBL. A test is made to determine if this is a duplexed data set and if the secondary RESTBL must still be processed. If both are true, REQUEST PAGE loops back to the point of the test for "no pages requested", above, and repeats the procedure. If the data set is not duplexed or if the secondary copy has been processed, a normal return is made to the caller.

If Request Page finds that there are not sufficient pages assigned, it attempts to assign sufficient pages from the secondary space allocation (the user may have requested up to 256 pages). If no such allocation was made, a call is made to ABEND. If such allocation was made, the RESTBL is examined to see if there is enough available space in the RESTBL to hold the new external page entries. This space exists in the RESTBL between the existing entries and the headers which are at the end of the RESTBL. If the RESTBL does not contain enough space and the data set is shared, a return is made to the caller with an error return code. If the data set is not shared, a call is made to the Expand RESTBL routine to provide the space required. Following this call or if the RESTBL contained sufficient free space, a call is made to Find External Page (CZCEL). Find External Page will return the addresses of 256 pages to be assigned, or the remainder of the secondary allocation if it is less than 256 pages. If the data set is not shared, this list is moved directly to the RESTBL following the last previous entry. If the data set is shared, the entries are moved, one by one, from the return area to the RESTBL, and each entry is preceded by a full word of zeros which serves as the interlock word for the page. For both types of data set a switch is set to indicate that the DSCB should be updated, and the number of available pages is updated in the RESTBL (RHDNEP). Request Page then returns to check if enough pages have been assigned. This procedure is repeated until enough pages are made available to the task or until the secondary space allocation is depleted. In the latter case a call is made to ABEND.

When the request for pages is satisfied, each page entry is flagged "in use" and the next available page location (RHDNAP) is adjusted. Finally, the checks for duplexed data set and secondary RESTBL update are repeated. If a secondary RESTBL must be processed, control is returned to the point of check for no pages; otherwise, a return is made to the caller with a normal return code.

Reclaim Routine (CZCOG)

Reclaim is called by Delete Page (CZCOD), GETNUMBR (CZCOO), and Stow (CZCOF) to delete external page entries from the "in use" list in the RESTBL and place them in the available list. This has the effect of removing the pages from the data set, but the pages remain allocated to the task and may be reassigned at some future time. (See Chart KE.)

Attributes: Read-only, public, privileged.

Entry Point: CZCOG1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB. Fields used in the DCB are:

DCBNI -- First page to be deleted (partioned data set).

DCBN -- First page to be deleted (nonpartitioned data set).

DCBM -- Number of pages to be deleted.

Module Called: ABEND (CZACP1) -- Abnormal task termination.

Exits:
Normal -- Return to the calling routine.

Error -- A return code of '04' is set in register 15 if the request is to delete shared pages which are interlocked. ABEND is called if the data set organization is VPAM.

Operation: The operation of Reclaim is essentially the same for shared and non-shared data sets. In the case of duplexed data sets, the process is repeated for the secondary RESTBL in order to maintain symmetry between the copies.

The function of Reclaim is to mark the deleted pages not in use and to add them to the available list in the RESTBL. This is accomplished by setting the proper flags in the external page entries in the RESTBL, and by moving the external pages entries from the chain of "in use" entries to the

chain of available entries. This creates a gap in the chain of all entries assigned to the task, so Reclaim moves the entries up in the RESTBL to close the gap (see Figure 21). In addition, the integrity bit in the RESTBL header is set. This causes CLOSEVAM (CZCOB) to call Write DSCB which writes out the entries from the RESTBL to the DSCB on the volume. The effect is to make all deleted pages which are not reassigned to the task, available for allocation to some other task after the data set is closed.

When Reclaim is entered, registers are saved in conformance with linkage conventions and base registers are established for the CSECT, PSECT, DCB, DCB header, and the RESTBL header. The number of pages to be deleted is checked and, if zero, control is returned to the user. If there are pages to be deleted, the address of the first external page entry to be deleted is computed, together with the amount and address of unused space in the RESTBL.

At this point, the processing for shared and nonshared data sets diverges. The distinction arises from the difference in the size of the external page entries (8 bytes for shared, 4 bytes for nonshared). The processing is logically identical for both and the following description applies to both.

The amount of available space in the RESTBL is compared to the amount of space required to hold the deleted entries. If the space is large enough, the deleted entries are moved from the "in use" chain to that area, the entries are flagged as not in use, and the entries following the gap left by the deletion are moved up to close the gap.

If the amount of available space is not large enough, a 64-word work area in the PSECT is used for intermediate storage of the entries. Up to 32 shared or 64 non-

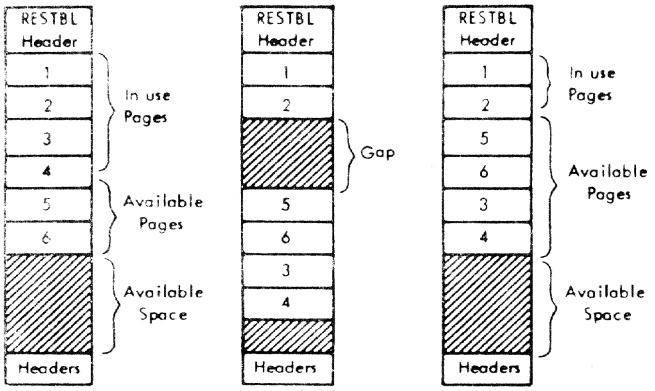


Figure 21. Deleting Pages from the "In Use" List in RESTBL

shared entries are moved from the "in use" chain to the work area, and are flagged not in use. The gap left by this deletion is closed and the deleted entries are added to the end of the chain of available entries. This process is repeated until all deletion entries have been processed.

Following the movement and flagging of entries, the data set is checked to see if it is duplexed and if the secondary RESTBL remains to be processed. If both conditions are met, the entire process is repeated for the secondary RESTBL. When all entries for all copies of the data set have been handled, the DSCB integrity bit (RHDFLG) in the RESTBL header is set, to cause the updating of the DSCBs on the volume or volumes, and control is returned to the caller.

DELVAM Routine (CZCFT)

Delete a VAM Data Set is called by Erase (CZAEJ), ADDCAT (CZCFA), and Recreate Public Storage (CZAXX) to delete a VAM data set by deleting its catalog entry and marking all its DSCB and data page entries "available" in the page assignment table. If the data set is private, the volume or volumes on which it resides are demounted. Optionally, the associated JFCB may be deleted. (See Chart KF.)

Attributes: Read-only, public, privileged.

Entry Points:
CZCFT1 -- Main entry, entered via type-1 linkage.

CZCFT2 -- For generation data groups only, entered via type-1 linkage.

Input: On entry, register 1 contains a pointer to a three-word parameter list:

Word 1 -- If entered at CZCFT1, a pointer to a 35-character (without userid) data set name. If entered at CZCFT2, a pointer to a 44-character (includes userid) data set name.

Word 2 -- A pointer to the JFCB to be deleted if that option is chosen.

Word 3 -- A pointer to the JFCB deletion indicator. This indicator is a one byte flag which is set to X'00' if no deletion is to be done or to X'80' if the JFCB is to be deleted.

Modules Called:
FINDDS (CZAEC1 or CZAEC2) -- Locates or creates a JFCB for the data set when a pointer to one is not provided by the calling routine.

LOCFQN (CZCFL2) -- Get T-block to see if BULK I/O pending.

SETXP (CEAH7) -- Prepares the external page table to allow a virtual storage page to be read in.

Search SDST (CZCQE1) -- Provides a count of users currently sharing the data set.

RELEXPG (CZCEN1) -- Called to release all data pages and DSCB slots used by the data set and allocates them for future assignment on the volume.

DELCAT (CZCFD1) -- Deletes the catalog entry for the data set.

FREEMAIN (CZCHA3) -- Frees the space occupied by the private volume table.

RELEASE (CZAFJ3 or CZAFJ6) -- Release the JFCB when that option has been specified.

READWRIT (CZCEM) -- Read in DSCB page.

DSCBREC (CZCEF1) -- Attempt recovery on a read DSCB error.

WTL (CZABQ1) -- Send message to operator and record it in the system log.

Exits:

Normal -- Return to the calling routine with one of the following codes in register 15:

'00'	Data set deleted.
'04'	Deletion not made due to open DCBs.
'08'	No deletion due to active sharer.
'0C'	No parameter list passed or FQN not indicated by Word 1.
'10'	Shared data set not owned, not unlimited access.
'14'	DSORG not VAM.
'1C'	Delete option specified, with JFCB pointer 0.

Error -- Before continuing, a message is written to the system operator and system log via WTL under any of the following conditions:

- When a good return code is not received from RELEXPG.
- When an attempt to recover a DSCB with a bad checksum is unsuccessful.
- When end-of-DSCBs occurs before end-of-volume fields.

Operation: On main or secondary entry, DELVAM saves input registers and establishes base registers for the CSECT and PSECT in conformance with linkage conventions. The secondary entry, at CZCFT2, allows a generation data group flag to be turned on, and then joins the main logic. The input parameter list is now checked for the presence of fully qualified name. If none is present, an error return is made to the calling routine. If the FQN is present, a test is made for the presence of a JFCB address. If no such address is given, a call is made to FINDDS, to find a JFCB or to create one. For a generation of a generation data group data set, FINDDS is called at a secondary entry point, CZAEC2, using the userid specified in the input parameter list. Otherwise, FINDDS is called at CZAEC1 and task common will be referenced for the userid. Once the JFCB has been located or created, tests are made to determine if only one DCB is open and if the data set is VAM organization. If either test fails an error return is made to the calling routine.

If DELVAM was called by LOGOFF, ABEND, or Close, a call to LOCFQN is made to check for Bulk I/O pending. If it is pending, normal return is immediately made to the caller.

Next the data set must be tested for sharing. This is determined by a call to Search SDST. If the data set is shared, only one user may be currently accessing the data set and the user deleting the data set must be either the owner or a sharer with unlimited access. If either of these conditions is not met, an error return is made to the caller.

Once this testing is complete or if the data set is not shared, DELVAM checks to see if a RESTBL exists. If one does exist, the data set is still open and it must be closed, calling Release at CZAFT6. Once the DCB is closed or if it was closed to begin with, a test is made to see if there is a DSCB. If none exists, there are no pages in the data set and processing will continue with the JFCB release described below.

Data Set Page Release: When the data set is deleted, all external pages occupied by the data set are freed for future allocation to another task. If the data set is private, a private volume table pointer is stored in the caller's PSECT. After this or if the data set is on public storage, a dummy RESTBL is constructed. This is necessary since the prior close of the DCB caused the deletion of the RESTBL. Each

DSCB is then read in turn. For private volumes there may be several volumes in the volume list. These entries are skipped over until the external page entries are found. These external page entries are then moved to a parameter list area in preparation for a call to RELEXPG. When the parameter list is full or when all external page entries have been moved to the list, RELEXPG is called to perform the release. The pages are marked available for assignment in the page assignment table. This procedure is repeated until all external pages have been released. If, in the process of releasing pages, an entire DSCB page is cleared of DSCBs, the DSCB page is also added to the parameter list and it is released. Any errors encountered in RELEXPG result in messages to the system operator and log.

JFCB Release: Following the release of external pages, the pointer to the dummy RESTBL is cleared. Then, or if the data set had no pages, the JFCB deletion indicator is tested. If no deletion is requested, processing continues with Catalog Deletion below. The JFCB is released by calling Release. An error return from that routine results in messages to the system operator and log.

Catalog Deletion: If the catalog entry is to be deleted, a call is made to DELCAT to perform the deletion. If an error return is received from DELCAT, the operator is notified and it is recorded on the system log via WTL. The catalog entry will not be deleted if the data set had no external pages, if DELVAM was entered by the Recreate Public Storage routine, or if the data set is USERCAT or SYSCAT. Following the deletion of the entry or if no deletion is performed, the exit procedure is entered. This process involves testing to see if the data set is public or private. For private data sets, a call is made to FREEMAIN to release the virtual storage occupied by the private volume table. DELVAM then returns to the calling routine by means of the RETURN macro instruction.

The data set sharing facility provided in TSS/360 requires that the data sets to be shared be organized for access by one of the virtual access methods. The sharing concept also relies on a system of interlocks which prevent the simultaneous reading and/or writing of one record by more than one authorized user. Two types of interlock are provided -- a read interlock and a write interlock.

A read interlock prevents a user from writing to a data set, data set member, or data set page. A read interlock is never actually placed on a data set. The interlock is implied by the presence of an entry for the data set in the shared data set table. When a user attempts to open a data set, the shared data set table is searched. If there is an entry in the table for the data set, the write interlock is tested. If that interlock is set, no other task may access the data set. If the write interlock is not set, a read interlock is assumed and a second task may open the data set for any operation which would impose a read interlock. Several users may therefore have simultaneous read access to a data set but if only one user has write access to it, no other user's task is allowed any access. Read interlocks are set on pages of a shared indexed sequential data set. This interlock prevents other users from writing to the interlocked page but permits them to read from it.

A write interlock prevents read or write access to shared data by any user other than the one who caused the interlock to be set. Only one write interlock can be set on a given data set at a time and once set, it precludes the setting of read interlocks on the data set. Additionally, if a read interlock is in effect on a unit of data, a write interlock cannot be set.

Interlocks are maintained in various tables depending on the data set organization. Regardless of the table which contains the interlock it is maintained in a four byte field called the interlock word. This field is aligned on a word boundary and has this format:

W - Write interlock

R - Read interlock

N - Number of interlocks currently set

I - Interlock for changing bytes R and N

A virtual partitioned data set is interlocked at the member level. The interlock word for each member of a partitioned data set is contained in the member header in the RESTBL (CHAMHD). The interlock is set according to the organization of the member and the option specified in the OPEN macro instruction. The interlock is set when the FIND macro instruction is issued, and released when the STOW macro instruction is issued or when the data set is closed. Table 34 lists the open options, and the type of interlock which is set for each. Shared access for members depends on the organization of the member as discussed below.

Virtual sequential data sets are interlocked at the data set level. The interlock word for these data sets is located in the data set entry of the shared data set table (CHASDE). The type of interlock imposed depends on the option specified in the OPEN macro instruction. Table 35 lists these options and the applicable interlock. The data set is interlocked when it is opened and the interlock is released when the data set is closed. Since several users may have simultaneous read access to a data set, a count is maintained of the number of users currently accessing the data set. As each user opens the data set, the count of users is incremented. As each user issues the CLOSE macro instruction, the count is decremented and only when it reaches zero is write access permitted.

A virtual indexed sequential data set or member is interlocked at the page level as well as at the data set level. That is to say, while one user is reading from one page of a data set, other users are restricted in their access to that page, but may access other pages in the data set assuming that the type of access attempted does not violate the data set or member level interlock. The page level interlock words for virtual index sequential data sets are contained in the external page entries (CHAEPE) in the RESTBL. One word of interlocks exists for each page of the data set and the locks are set and reset by the Movepage routine (CZCOC). Table 36 lists the effects of various open options on page level interlocks.

A page level read interlock is placed on a page when a GET or type-KY READ macro

Table 34. Effect of OPEN Option on Member Interlocks in Member Header

OPEN Option	Interlock Type (Set by FIND)
INPUT	READ
OUTPUT	WRITE
INOUT OUTIN UPDATE	WRITE (VSAM member) READ (VISAM member)

Table 35. Effect of OPEN Options on Data Set Interlocks in SDST

OPEN Option	VSAM	VISAM
INPUT	Implied READ	Implied READ
OUTPUT	WRITE	WRITE
INOUT OUTIN UPDATE	WRITE	Implied READ
Note: No interlocks are set on VPAM data sets at OPEN time.		

Table 36. Effect of OPEN Option of VISAM Page Level Interlock

OPEN Option	Interlock Type	When Set
INPUT	READ	When page is read
UPDATE	WRITE	READ Exclusive, Write by new key,
INOUT		WRITE replace by key, DELRAC
OUTIN	READ	READ by key and all other operations causing a page to be read
OUTPUT	None	

instruction is issued. A page level wirte interlock is set by the type-KX READ macro instruction which also releases a page level read interlock set by that task. Other macro instructions which release a page level read interlock when issued by the task which set it are WRITE, ESETL, DELREC, RELEX or CLOSE. If the task issues any macro instruction which references another page, the read interlock is released. A page level write interlock is released by the GET, type-KY READ, RELEX, WRITE, DELREC, or CLOSE macro instruction or by any macro instruction which references a page other than the one in which the write interlock is set.

CONTROL TABLE INTERLOCKS

VAM has two control tables which reside in shared virtual storage - the RESTBL and the SDST. Routines which manipulate these tables must interlock them to prevent them from being changed while in use. These interlocks do not have the conventional VAM interlock word format.

The SDST interlock is a one byte field at the beginning of the SDST control entry. While that lock is set, no other task may access the SDST. Within the SDST, data set and member entries are made to record the allocation of shared storage. The Search SDST routine, which makes the SDST entry, does not allocate the shared storage, but simply reserves space in the SDST for the allocation to be recorded. The reserved space must be interlocked to prevent system usage of the SDST entry before the allocation is actually completed. This is done by setting the SPT number to X'FFFF', which is an invalid SPT number. After allocation of shared storage, the routine obtaining the shared virtual storage will store the actual SPT number in the SDST, thus releasing the pseudo-lock.

The RESTBL interlock must be set by those routines manipulating the RESTBL of shared data sets. This includes most of the VAM modules. The RESTBL interlock is the first byte of the RESTBL header, and has the same effect as a conventional write interlock. Most VAM routines can be called at a variety of levels; that is, when a routine is called that may manipulate the RESTBL, the interlock may or may not already be set. For this reason, each routine attempts to set the lock. If the lock is already set, the tasks must wait until it is reset, unless it was set by the current task against the current DCB. This information is in the RESTBL. The ID of the task imposing the RESTBL lock is recorded in the RESTBL header (RHDTID), and the interlocks charged against a DCB are recorded in the DCB header.

As with setting the RESTBL lock, each routine which attempted to set the lock must attempt to reset it, but only the routine which actually set the lock will effect its release. This is accomplished by storing, in the RESTBL header, the PSECT location of the routine which caused the lock to be set. When backing out of a nest of calls, each routine will attempt to reset the RESTBL lock, but if it did not set it, a return is made to the next higher level with the interlock still in force.

Sharing of virtual organization data sets is dependent upon the owner permitting the sharing by means of the PERMIT command and the user declaring his intention to share the data set by means of the SHARE command. The command system routines and catalog service routines set indicators in the owner's catalog which indicate his willingness to share the data set, the users with whom he is willing to share it, and the access to the data set he wishes each to have. These routines also set indicators in the user's catalog which associate the data set name he is using with the owner's data set.

Three routines in the access methods provide the additional processing required to facilitate the sharing of data sets. The first two of these routines, Interlock (CZCOH) and Release Interlock (CZCOI), maintain interlocks in various tables in the system. These interlocks prevent two users from simultaneously updating a data set.

The third of these routines is the Search SDST routine (CZCQE). This routine searches the shared data set table in an effort to locate the name of a shared data set or member of a shared partitioned data set. This table consists of a group of chained data set or member entries, which correlate opened data sets or members with their respective shared page tables. These shared page tables, in turn, specify the location of each page of the data set in main storage, and are followed by the external shared page tables, which list the external storage addresses of the data set pages.

If a data set is not already listed, this routine will create an entry for it; if the data set is listed, the count of users sharing it is incremented. This routine is also used to delete entries or decrement the count of users sharing the data set, as each user closes his DCB associated with the data set. A description of Search SDST is included in the preceding section.

Interlock Routine (CZCOH)

Interlock is called by other system routines to impose read (R) or write (W) interlocks on an interlock control word. The type of interlock set depends upon the OPEN option. Tables 34, 35, and 36 summarize the effects of these options on the operation of Interlock. The interlock control words are used to control shared access to the POD, RESTBL (header or external page entries), member headers, and SDST data set entries. When a write interlock is imposed, no additional read interlocks may be set. If a read interlock is set, the task attempting to set a write interlock will wait until all read interlocks have been removed by the tasks that set them. No additional read interlocks will be set during this wait. (See Chart LA.)

Restrictions: It is impossible to impose more than 255 read interlocks with this routine.

Attributes: Read-only, reenterable, privileged, public, system.

Entry Point: CZCOH1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the following parameter list:

Word 1 -- Address of the interlock control word.

Word 2 -- Address of the type code specifying the interlock to be set.

Word 3 -- Address of the DCB associated with the data set on which the lock is being imposed.

The type code pointed to by word 2 is a 2-byte field described as follows:

Byte 1 specifies the type of interlock to be reset:

C'R' = Read
C'W' = Write

Byte 2 specifies the table in which the interlock is to be reset:

X'00' = SDST
X'04' = RESTBL
X'08' = POD
X'0C' = Member Header
X'10' = External Page

The DCB address normally specified in word 3 may be set to zero if it is not known. However, during the time that an interlock remains imposed after no DCB address was specified, it must be assured that the system will not ABEND.

Modules Called:

TSEND (CEAP9) -- End this task's time slice while waiting for a lock to be reset by another task.

SYSER (CEAIS2) -- Report invalid parameters as input.

ABEND (CZACP1) -- Terminate task after SYSER.

XTRCT (CEAH03) -- Get task ID for RESTBL header area.

Exits:
Normal -- Exit is to the calling routine with the specified interlock set. No completion code is given.

Errors -- The type codes supplied are checked for validity and a SYSER and ABEND are given if either is invalid.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the CSECT and PSECT, parameter list, DCB, DCB header, interlock word and type codes.

The interlock type and the table type where the interlock is to be imposed are tested for validity.

The write interlock byte of the interlock word is tested to determine if a write interlock is already in force. If the write interlock is on and the request is not for a RESTBL lock, time slice end is forced by calling TSEND. This allows other tasks to continue processing and eventually release the interlock that the current task was attempting to set.

The RESTBL lock is handled slightly different from other interlocks. Each routine which manipulates the RESTBL attempts to set a write interlock on the RESTBL, which is accounted for on a DCB within task basis. A call to the interlock routine may be executed for a RESTBL interlock and the RESTBL may already be locked. If this condition occurs and the same task which set the initial lock is attempting to set another against the same DCB, control is returned to the caller, as if the lock had just been set. Otherwise, TSEND is called.

If the write interlock byte of the interlock is not set, interlocks may be set as desired. The locks will be recorded in the interlock summary word of the associated DCB header. In addition, if the lock is set on the RESTBL, the virtual storage address of the PSECT of the calling routine is saved in the interlock word, and is used in the interlock reset process.

The interlock to be set is either read or write. The write lock which is already set is recorded in the DCB header, and control is returned to the caller by the RETURN macro instruction if no read locks are currently set. Otherwise, TSEND is called to wait for the read locks to be released.

Read locks are set cumulatively. The read interlock is set and the read interlock counter is incremented to indicate the presence of one or more read interlocks. This read interlock process is controlled by an additional lock byte within the interlock word itself. The write lock is reset, the read lock is accounted for in the DCB header, and control is returned to the caller by the RETURN macro instruction.

Release Interlock Routine (CZCOI)

Release Interlock (RLINTLK) is called by system routines to release read (R) and write (W) interlocks on an interlock control word. (See Chart LB.)

Attributes: Read-only, reenterable, privileged, public, system.

Entry Point: CZCOI1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of a 3-word parameter list as follows:

Word 1 -- Address of interlock word.

Word 2 -- Address of the type interlock.

Word 3 -- Address of DCB being processed.

The type of interlock pointed to by word 2 of the parameter list is a 2-byte field described as follows:

Byte 1 specifies the type of interlock to be reset.

```
C'R' = Read
C'W' = Write
```

Byte 2 specifies the table in which the interlock is to be reset:

```
X'00' = SDST
X'04' = RESTBL
X'08' = POD
X'0C' = Member Header
X'10' = External Page
```

The DCB address normally specified in word 3 of the parameter list may be set to zero, if it is not known. It should be known, if it was known at the time the lock was set. If it is not specified the interlock summary bits in the DCB header will not be reset.

Modules Called:
TSEND (CEAP9) -- End time slice while waiting for a lock to be reset by another task.

SYSER (CEAIS2) -- Invalid input to CZCOI.

ABEND (CZACP1) -- Terminate task after SYSER.

VDMEP (CZCQK) -- When desired lock to be set is not set.

Exits:
Normal -- Exit is to the calling routine by the RETURN macro instruction. No completion code is given.

Error -- The type codes are checked for validity and SYSER and ABEND are called if either is invalid.

VDMEP is called if an attempt is made to release a lock that is not set.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for CSECT and PSECT, parameter list, DCB, DCB header, interlock word and type codes.

If the table type code specified is invalid, SYSER is called. Also, if the DCB is specified, and the summary bit in the DCB header corresponding to the interlock intended to be released does not indicate that the lock is on, VDMEP is called.

A release request may be for a read or a write interlock. A special case of this is the RESTBL interlock. The RESTBL interlock must be set and must be released by the same routine that set it. The PSECT address of the routine requesting reset is compared to the PSECT address stored in the interlock word. If they are the same, the lock is reset and control is returned to the caller by the return macro instruction. Otherwise the lock is not reset before returning.

A request to release a write interlock will cause the interlock to be reset. If the DCB is specified and the summary does not indicate that the requested lock is set, SYSER is called. If the summary does include the requested lock, the bit is cleared and control is returned to the caller by the RETURN macro instruction.

Read interlocks are released by decrementing the read interlock counter. When the counter goes to zero, the physical and read interlock is cleared.

The manipulation of the read interlock and the read interlock counter are controlled by an additional lock in the interlock word. The control lock is reset and the summary bits are reset if a DCB was specified.

SECTION 4: OPEN AND CLOSE PROCESSING

This section describes the routines which prepare a data set and its related control blocks for processing (OPEN) and routines which remove a task's references to a data set and release unneeded space when a task no longer wishes to reference the data set (CLOSE). The routines in this section can be entered as a result of one of seven macro instructions being issued:

1. OPEN -- a user wishes to access or create a data set.
2. DUPOPEN -- a user wishes to access or create a duplexed data set.
3. FIND -- a user wishes to access a member of a partitioned data set.
4. CLOSE -- a user has finished using a data set.
5. DUPCLOSE -- a user has finished using a duplexed data set.
6. STOW --a user has finished using a member of a partitioned data set.
7. ABEND -- a task is being abnormally terminated and its data sets must be closed.

Figures 22 and 23 depict the interaction among the various modules involved in opening and closing a data set. The routines Open Common (CZCLA) and Close Common (CZCLB) contain the initial point of linkage from the OPEN and CLOSE macro instructions for all access methods and have been described earlier in this manual.

Also included in this section is a description of the VAM ABEND Interlock Release routine. When a task is abnormally terminated because of some error condition, the close processing is performed to close any data sets that the task may have been using. The VAM ABEND Interlock Release routine is called to locate and to release any interlocks which were set when the ABEND was issued.

OPEN PROCESSING

The routines associated with Open processing include OPENVAM, DUPOPEN, VSAM Open, and VISAM Open.

OPENVAM Routine (CZCOA)

OPENVAM is called by Open Common (CZCLA) to perform DCB and data set initialization common to all of the virtual access methods. (See Chart MA.)

Attributes: Closed, reentrant, privileged.

Entry Point: CZCOA1 -- Entered via type-1 linkage.

Input: Register 1 contains a pointer to a four word parameter list as follows:

Word 1 -- Address of the DCB.

Word 2 -- Address of the JFCB.

Word 3 -- Address of "Fence Straddler" save area.

Word 4 -- Address of the public or private volume table (PVT).

Modules Called:

Connect (CZCG7) -- Connect task to shared RESTBL

ESA Lock (CZCEJ1) -- Lock PAT. (CZCEJ2) -- Unlock PAT.

REQPAGE (CZCOE1) -- Assign directory page (VPAM).

RELEXPG (CZCEN1) -- Release pages.

DELCAT (CZCFD1) -- Release catalog entry.

FREEMAIN (CZCGA3) -- Free virtual storage.

Search SDST (CZCQE1) -- Creates a shared data set table entry for a newly opened data set or increments the count of users for the data set if it is already open.

Interlock (CZCOH1) -- Interlocks the RESTBL of a shared data set when the routine is operating on it.

Release Interlock (CZCOI1) -- Releases the RESTBL interlocks placed on shared RESTBLs.

DSCB READ/WRIT (CZCEM1) -- Request a PAT write.

VDMEP (CZCQK1) -- Issue diagnostic message and terminate function but not task.

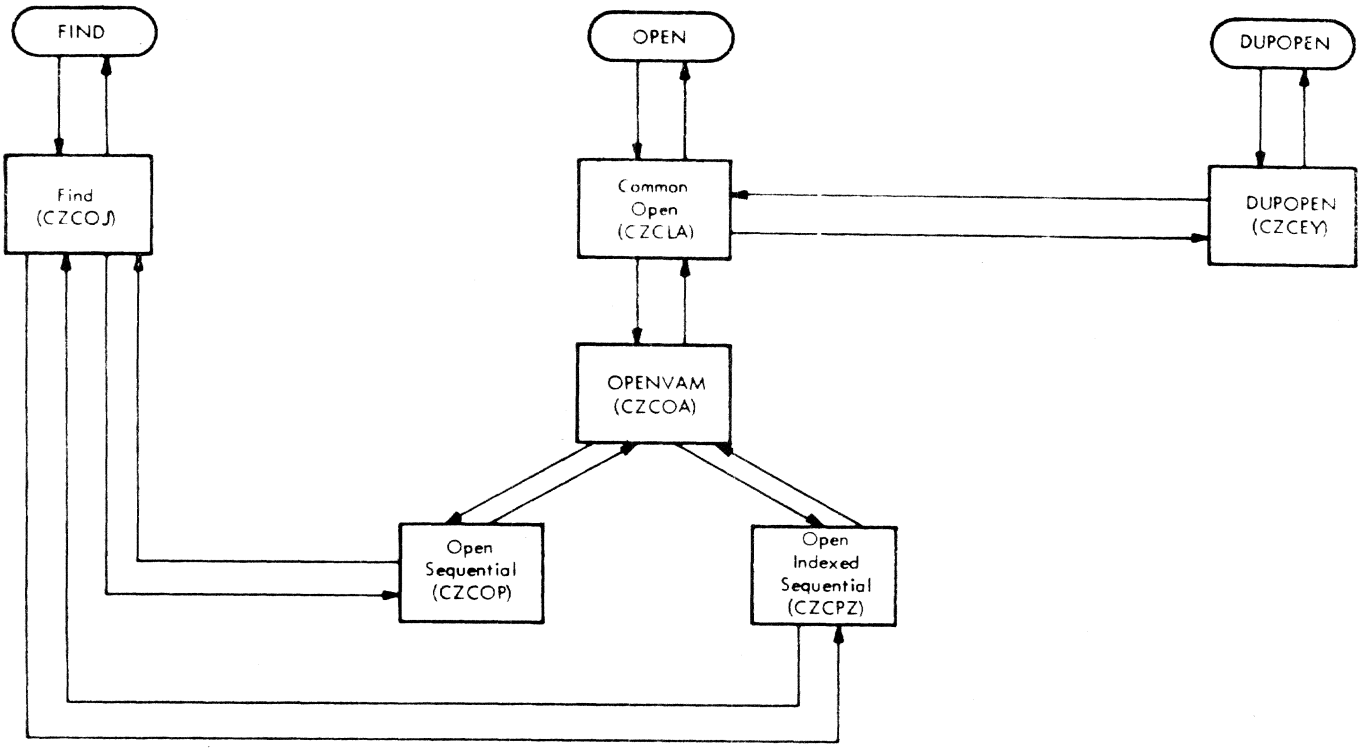


Figure 22. Module Interaction in VAM Open Processing

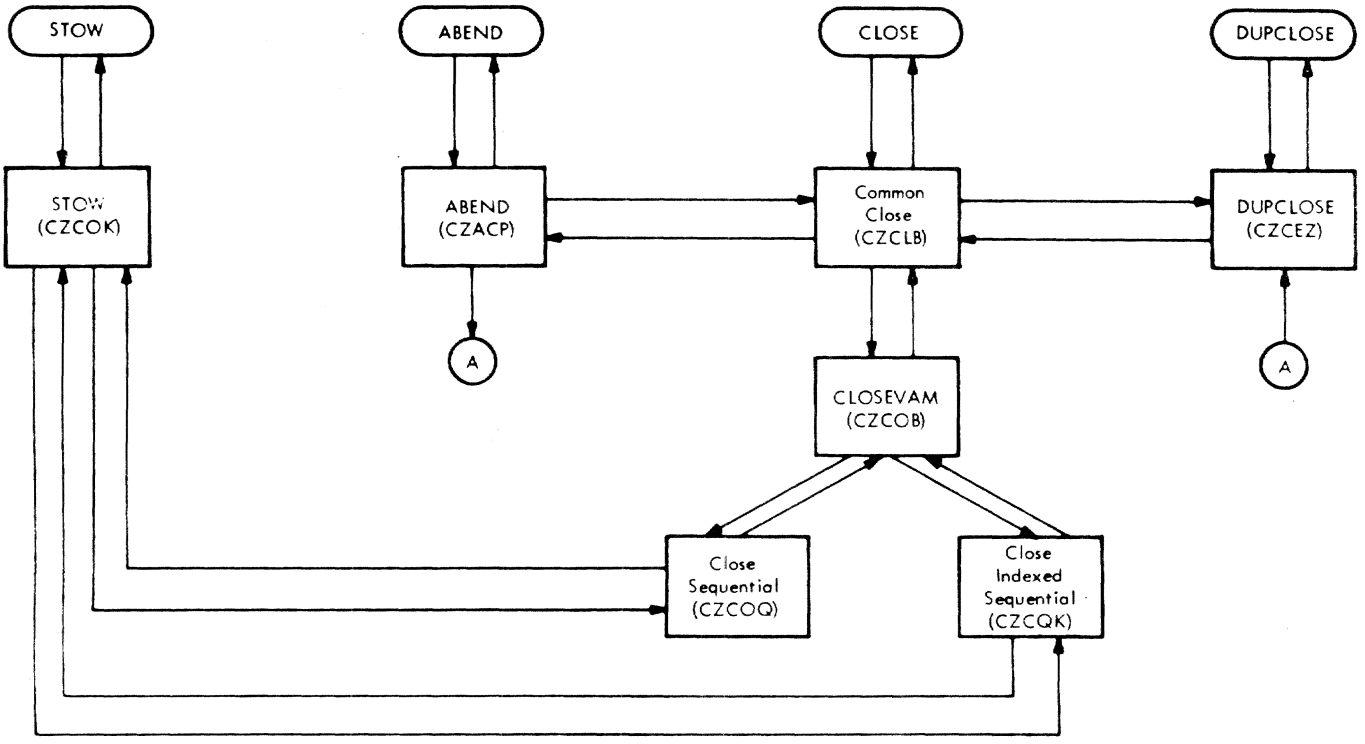


Figure 23. Module Interaction in VAM Close Processing

DSCB RECOVERY (CZCEF1) -- Attempts to recover from an error when the DSCB is read.

GETMAIN (CZCG2) -- Obtains virtual storage for nonshared RESTBLs and for required directory pages for nonshared data sets.

GETSMAIN (CZCG6) -- Obtains shared virtual storage for shared RESTBLs.

FINDEXPG (CZCEL1) -- Assigns external storage space to a new data set.

WRITDSCB (CZCEW1) -- Writes out updated DSCBs to the volume.

CKCLS (CEAH24) -- Obtains the protection class of the task.

XTRCT (CEAH03) -- Obtains the ID of the task.

Expand RESTBL (CZCQI1) -- Increases the size of the nonshared RESTBL when necessary to contain new headers.

MOVEPAGE (CZCDC1) -- Reads in the directory pages of a partitioned or indexed sequential data set.

VSAM OPEN (CZCOP1) -- Performs open processing unique to sequential data sets.

VISAM OPEN (CZCPZ1) -- Performs open processing unique to indexed sequential data sets.

Exits:
Normal -- Return to calling routine.

Error -- VDMEP is called under the following conditions:

- Error encountered trying to PGOUT updated DSCB.
- DSCB has invalid chain field.
- Attempt to expand shared data set.
- Relative volume number is out of volume table limit.
- Bad return code from CZCEM; failure to read DSCB.
- An attempt to open a non-VAM data set.
- Search SDST could not link to shared VM.
- Should be E-type DSCB but is not.
- Total pages used by data set in error.
- External page request exceeds amount available.
- Attempt to open new data set for read-only.
- Two active users using private shared data set.
- Unrecoverable DSCB error.
- DSCB does not contain correct number of extents.

Operation: On entry, OPENVAM saves all input registers and establishes base registers for the CSECT, PSECT, RESTBL, DCB, and JFCB. The user's authorization to open the data set and the data set organization are checked. If the user is not authorized to open the data set or if the data set is not VAM, VDMEP is called.

Two types of data sets are processed by OPENVAM, shared and nonshared. While the objective of OPENVAM is functionally similar for both types, that is, to prepare data sets for access, they are handled differently. The logic flow of OPENVAM diverges at specific points to accomplish processing for shared or nonshared data sets.

One of the functions performed by OPENVAM is to build a RESTBL for a new data set. Tests are made to see if the data set has already been opened. If the data set has been opened, a RESTBL already exists and this processing can be skipped. The tests for opened DCBs differ for shared and nonshared data sets. If the data set is shared, the task currently opening the data set may have opened it. In this case, the count of open DCBs in the JFCB is nonzero. If another task has opened the data set, an entry will exist in the shared data set table.

A call is made to Search SDST (CZCQE) to locate or to create an entry. Two return codes from CZCQE are acceptable. One code indicates that an entry existed. In this case, a call is made to Connect (CZCG7) to connect the task to the shared RESTBL which exists. The second code indicates that no entry existed but one was created. In this case, the RESTBL must be created since the data set is newly opened. Any return code other than these two is considered an error and VDMEP is called.

If the task currently opening the data set had DCBs already open for it or if the task was connected to the shared RESTBL, the following processing to create the RESTBL is skipped.

For nonshared data sets, the test involves a check of the JFCB to see if the count of open DCBs is nonzero. The same criteria apply to nonshared data sets as to

shared. If DCBs are open, no new RESTBL is built.

Building the RESTBL: For all newly opened old data sets, the format-E DSCB is read into virtual storage. This is accomplished by means of the SETXP macro instruction which simply adds the external page number to the external page table for the task. Any reference to the DSCB page will result in a paging operation by the resident supervisor. Once the DSCB page is in virtual storage it is checked for errors. A checksum error results in a call to DSCB Recovery (CZCEF) to attempt recovery. A bad volume pointer, a pointer to a non-DSCB page, a bad relative page number for a device, or the case where data set names in the JFCB and the DSCB do not match, result in a VDMEP call.

Once the format-E DSCB has been read in, the number of pages required to contain the RESTBL is computed. For new data sets, this number is a function of the primary space allocation as indicated in the JFCB. For existing data sets it is a function of the size of the data set as it exists. If the data set is shared, this RESTBL size is increased by 2 pages. If the data set is SYSLIB, 10 pages are added; for SYSCAT, 5 pages are added. This increment is used because a shared RESTBL cannot be expanded and the value 2 represents a "best guess" of the ultimate size of the RESTBL. The value should be adjusted as installation needs dictate. If the shared data set is also index sequential, an additional 2 pages are used.

After the RESTBL size has been computed, a call is made to GETMAIN or to GETSMAIN to obtain virtual storage for the RESTBL. When virtual storage has been obtained for the RESTBL, it is interlocked, if shared, and the RESTBL is filled in. This process starts with the building of the RESTBL header. The total number of pages is stored; if the data set is not new the counts of overflow, directory, and data pages are stored, and, for all data sets, the number of assigned pages is compared to the number of "in use" pages. If more pages are in use than are assigned, VDMEP is called. If no error exists, the RESTBL is chained to the JFCB and pointers to the format-E DSCB and the volume table are placed in the RESTBL.

Next, the data set is checked to see if it is new. If so, a call is made to FINDEXPG (CZCEL) to get external pages for assignment to the data set, and the pages returned are placed in the RESTBL and marked assigned but not in use. A call is later made to WRITDSCB (CZCEW) to place the updated entries in the DSCB chain. If an error return is received from FINDEXPG, VDMEP is called. For existing data sets, a test is made to see if the volume is public or private. In the case of private volumes, a list of private volumes is contained in the DSCBs before the page entries; these volume IDs must be skipped and are not placed in the RESTBL. For existing data sets, the calls to FINDEXPG and WRITDSCB are not required.

When the external page entries from the DSCB, or for new data sets from the FINDEXPG routine, are placed in the RESTBL, a distinction is made between shared and nonshared data sets. For nonshared, the entries are simply moved into the RESTBL. For shared data sets, an interlock word must be provided before each page entry. This interlock word is inserted and set to zero as each entry is placed in the RESTBL.

When the RESTBL has been built, the DCB header must be built and added to the chain of headers in the RESTBL. This last step is performed first. The available space in the RESTBL is checked to see if there is space to contain the new header. If no space is available, a call is made to Expand RESTBL to increase the size of the RESTBL. This is only done for nonshared data sets. If the condition occurs for a shared data set, a call is made to VDMEP. When space has been found to contain the header, it is chained to the list of headers in the RESTBL and the DCB header is then built.

Building the DCB header involves chaining it to the DCB, the RESTBL, the RESTBL header, and the JFCB. The protection class of the task is determined by a call to CKCLS (CEAQ4), and the task ID by a call to XTRCT (CEAH2); these values are stored in the DCB header. The VAM read-only flag in the JFCB, if turned on by Common Open, will be turned off and the read-only access flag turned on in the DCB header. Finally, the address of the directory or the POD is placed in the DCB header.

If the data set is shared, there are no other DCBs open, and the data set is either partitioned or index sequential, it may be necessary to assign directory pages. If no directory pages exist for the data set, a call is made to Request Page (CZCOE) to assign the first page as a directory page. After this or if directory pages existed, a call is made to MOVEPAGE to read the directory pages into virtual storage, and the number of directory pages is placed in the RESTBL. Next, or if the data set was partitioned or sequential, or if the data set was already opened, the access dependent open routine is called as described below.

For nonshared data sets the processing is essentially the same with two excep-

tions. First, a call is made to GETMAIN for virtual storage space prior to the call to MOVEPAGE. This is done to provide space for the directory pages. Second, the count of directory pages is not updated in the RESTBL and the access dependent open routines are called.

Access-Dependent Open Processing: The data set is now tested to determine its organization. If the data set is sequential, VSAM OPEN (CZCOP) is called; if it is index sequential, VISAM OPEN (CZCPZ) is called. These routines perform open processing unique to the data set organization. If the data set is partitioned, a member header is built and added to the chain of member headers in the RESTBL. This is done in the same manner as the adding of the DCB header with the call to Expand RESTBL for nonshared data sets, when necessary, and the call to ABEND when the RESTBL cannot contain the header for a shared data set. Once the member header has been built, it is linked to the DCB header, the RESTBL interlock is released, if the data set is shared, and control is returned to the calling routine.

DUPOPEN Routine (CZCEY)

The function of DUPOPEN is to open duplicate data control blocks for a duplexed data set residing on public volumes. One copy of the data set is the primary copy. The secondary copy resides on a separate public device and is, at all times, an exact copy of the primary copy. As such, the secondary copy can be used for recovery purposes when a read error occurs on the primary copy.

DUPOPEN flags the JFCBs (TDTDC1) to indicate the primary and secondary copies, and chains the two JFCBs by placing, in each JFCB, a pointer to the other (TDTDUP). By calling Open Common and, subsequently, OPENVAM, DUPOPEN builds a RESTBL for each copy of the data set and opens the DCBs. The address of each RESTBL is placed in its corresponding DCB and the field DHDDUP of each DCB header is set to point to the other RESTBL. (See Chart MB.)

Attributes: Closed, fetch protected, privileged, reentrant, nonrecursive, and residing in public virtual storage.

Entry Point: CZCEY1 -- DUPOPEN is entered by a type-1 or type-2 linkage generated by the expansion of the DUPOPEN macro instruction.

Input: Register 1 contains the address of the following parameter list:

Word 1 -- Address of the primary DCB.

Word 2 -- Address of the secondary DCB.

Word 3 -- Address of an option byte which contains:

bits 0-3 Not used
bits 4-7:

0000	Input
1111	Output
0001	Read Back
0011	Inout
0111	Outin
0100	Update

Modules Called:
OPEN COMMON (CZCLAO) -- A single call is made to open both the primary and secondary DCBs.

SYSER (CEAIS2) -- Minor system error with message.

ABEND (CZACP1) -- Abnormal task termination.

FINDJFCB (CZAEB) -- To create a JFCB when one is not found for the DCB being duplicated OPEN.

Exits:
Normal -- Return to the calling routine.

Error -- ABEND is called under the following conditions:

- No JFCB could be found or created for the primary or secondary DCBs.
- Data set not public.
- DCBs not compatible.
- Data set not VAM organization.
- Same JFCB specified for both DCBs.
- JFCB specified has been found in JOBLIB.

SYSER is called before each ABEND.

Operation: When DUPOPEN is entered, register 1 contains a pointer to a parameter list which, in turn, contains pointers to the primary and secondary DCBs. DUPOPEN, after performing the usual register save and base register initialization, saves the addresses of these DCBs and the open option in its own PSECT. DUPOPEN then begins to scan the list of JFCBs in the TDT beginning with the last JFCB created and working back toward the first.

For each JFCB it encounters, it compares the data set name in the JFCB with that in the DCB. If no matching JFCB is found in the entire list of JFCBs, FINDJFCB is

called to help create one. When the JFCB for the first DCB is found, its address is saved and the same procedure is repeated for the second DCB.

When both JFCBs have been located and tested to ensure they are not the same JFCB, they are examined to ensure that the corresponding data sets are on public storage and are VAM organization. If either data set fails either test, the task is terminated and is sent the appropriate error message.

A test is made to ensure that the DCB is not already in the user's JOBLIB.

Next, a test is made to see if any DCBs have been opened for the data set. If none has, the JFCBs are chained together and a flag is set in each to indicate which is the primary and which is the secondary. When this is done or if some DCB had already been opened for this data set, a parameter list is constructed for OPEN COMMON. This parameter list contains the addresses of both DCBs and the open option indicator. OPEN COMMON is then called to open both DCBs and is passed the address of the parameter list.

On return, the two new DCBs are compared to ensure that they match. If they do not match, ABEND is called and a message is sent to the task. If the DCBs do match, DUPOPEN links the corresponding RESTBLs and returns to the caller.

VSAM Open Routine (CZCOP)

VSAM Open is called by OPENVAM (CZCOA) and by FIND (CZCOJ) to initialize the data set buffers and the DCB to allow processing of a virtual sequential access method (VSAM) data set (See Chart MC.)

Attributes: Read-only, reenterable, privileged, public.

Entry Point: CZCOP1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB to be opened.

Modules Called:
GETMAIN (CZCG2) -- Obtain virtual storage space for the sequential buffer to be associated with this DCB.

SETL (CZCOT1) -- Initialize data set to begin processing at beginning or end.

Exits:
Normal -- Return to the calling routine.

Error -- VDMEP is called under the following conditions:

- Record length is specified greater than 1 segment (1,048,576 bytes).
- Record format code is in error, that is, not 'V', 'E' or 'U'.
- Record format is undefined, and record length is not a multiple of 4096 bytes.

Operation: General registers are stored in conformance with linkage conventions and base registers are declared for the PSECT and CSECT, a DCB, DCB header and a RESTBL.

The V-cons for the sequential user macros GET, PUT, PUTX, and SETL are stored in the DCB. These sequential routines are "fence sitters." Their PSECTS cannot be used for storage (save areas), therefore, save areas are allocated dynamically. Open Common gets a "scratch" page for this purpose and its location is stored in the GET R-con location in the DCB by OPENVAM. VSAM Open takes that value, computes save area locations for the other macros and stores the computed values in the respective R-con locations in the DCB.

VSAM Open is responsible for establishing input/output areas in virtual storage for the sequential access method. The space requirement is computed from the logical record length declared in the DCB (DCBLRE) at OPEN time. The maximum allowable record length is 1,048,576 bytes (1 segment); if this value is exceeded, ABEND is called to terminate the task.

The sequential access method processes three types of record format - fixed, variable and undefined. Special considerations are applied to each format in the allocation of I/O space.

Undefined records must have page-multiple record length. ABEND is called if the record length (DCBLRE) is not in page multiples.

Variable-length records have eight bytes of space control information appended to each record (four bytes preceding the record, provided by the user, and four bytes following, provided by VSAM). This adjusted value record length plus control bytes is used to compute the actual amount of storage required.

For fixed-length records whose record length is a page, only one I/O page is needed.

For variable-length and fixed-length records other than page multiples, a minimum of two I/O pages is required. GETMAIN is called by the GETMAIN macro to obtain the virtual storage to be used as I/O

areas. The virtual storage location of this area is stored in the DCB and the DCB header.

The record length (DCBLRE) is saved in the DCB header (DHDMRL) as the maximum logical record length. If this value is ever exceeded, and the condition can be detected, ABEND will be called. This could happen while processing undefined or variable-length records where the record length is defined dynamically by the user.

VSAM Open is also responsible for the initial positioning of the data set. A call is made to SETL (CZCOT) to set DCBLPA to the appropriate value. The DCB open option (DCBOP1) is used to determine what that value should be. For input, update, or INOUT, the data set is logically positioned to the beginning. For output or OUTIN the data set is positioned to the logical end.

Control is returned to the caller by the RETURN macro.

VISAM Open Routine (CZCPZ)

VISAM Open is called by VAM general services Open and by Find, to initialize DCB and page buffers. (See Chart MD.)

Attributes: Read-only, reenterable, public, privileged.

Entry Point: CZCPZ1 -- Entry to VISAM Open is by type-1 linkage (privileged to privileged).

Input: Register 1 contains the address of the DCB.

Modules Called:
GETMAIN (CZCGA2) -- Obtain page buffers for data and overflow pages.

SETL (CZCPC2) -- Position data set according to OPEN option.

Exits:
Normal -- Return to the calling routine.

Error -- ABEND is called if key offset + key length is greater than record length. VDMEP is called under any of the following conditions:

- Key length is less than the minimum.
- Record length exceeds maximum.
- Key offset goes beyond record.
- Overflow pages greater than 240, or directory pages greater than 255.
- Data set opened for Input or Inout and no data pages.

Operation: General registers are stored in conformance with linkage conventions, and base registers are declared for the CSECT and PSECT, a DCB, DCB header and a RESTBL.

The V-cons for the index sequential user macros GET, PUT, and SETL are stored in the DCB. These index sequential routines are "fence sitters." Their PSECTS cannot be used for storage (save areas), therefore, save areas are allocated dynamically on a DCB basis. Open Common gets a "scratch" page for this purpose, and its location is stored in the GET R-con location by OPEN-VAM. VISAM Open takes that value, computes save area locations for the other macros and stores the computed values in the respective R-con locations in the DCB. The PUTX macro is not used by VISAM Open. Its R-cons and V-cons are zeroed to prevent any erroneous usage.

Legality checks are made on DCB information-key length (DCBKEY), record length (DCBLRE), and relative key position (DCBRKP).

The minimum key length is one byte.

The maximum logical record length of a VISAM data set is 4000 bytes.

The origin of the relative key position must be within the range of the logical record length. Also, the relative key position plus the key length must not exceed the logical record length.

DCB fields peculiar to VISAM are initialized.

Data records may never cross page boundaries in index sequential organization, therefore, only one virtual storage page is required as an I/O buffer. This is obtained by the GETMAIN macro.

Where this is not the first DCB open for a nonshared data set within the same task, the DCB will be linked to the existing page buffer, and no call to GETMAIN is necessary.

If overflow pages exist, a virtual storage page is also needed as an overflow page I/O buffer. The maximum number of overflow pages is 240.

Having obtained all necessary virtual storage space, VISAM Open will logically position existing data sets (that is, RHDDAT=0). A nonexistent data set may not be opened for input, update or inout OPEN options. If there are no data pages, VISAM Open returns to the caller by the RETURN

macro. If the data set has data pages, a SETL (CZCPC) is executed as defined by the OPEN option. Input, update and inout cause a SETL to the beginning of the data set. Output and outin cause a SETL to the end of the data set. Control is then returned to the caller by the RETURN macro.

CLOSE PROCESSING

The Close routines consist of CLOSEVAM, DUPCLOSE, VSAM Close, VISAM Close, and VAM ABEND Interlock Release.

CLOSEVAM Routine (CZCOB)

CLOSEVAM is called by Close Common or by DUPCLOSE to perform close processing for a given DCB. It may also be called by ABEND to close the DCB for a task which is being abnormally terminated or by VDMEP if a function (but not the task) is being terminated. The major functions of CLOSEVAM are to:

- Delete only the DCB header for data sets which have other DCBs open.
- Release assigned but not used external pages of the data set by calling RELEXPG. This step is omitted if the HOLD option has been specified.
- Update the DSCBs on the volume by calling WRITDSCB.
- On the close of the last DCB open for the data set, the RESTBL and, if the data is partitioned, the POD are deleted by calling FREEMAIN.
- Delete the data set, its catalog entry, and its JFCB if "delete at close" is specified. This is accomplished by calling DELVAM.
- If the data set resides on a private volume and the last open DCB is being closed, the volume table is deleted by calling FREEMAIN. (See Chart ME.)

Restrictions: CLOSEVAM may not be called if the specified DCB is not open.

Entry Points:
CZCOB1 -- Normal entry via type-1 linkage.

CZCOB2 -- Direct entry from close command.

Input: Register 1 contains the address of the DCB being closed.

Modules Called:
Interlock (CZCOH) -- Interlocks the RESTBL for shared data sets.

Search SDST (CZCQE) -- Deletes the shared data set table entry for the task closing the DCB.

VSAM Close (CZCOQ) -- Performs close functions unique to sequentially organized data sets.

VISAM Close (CZCQA) -- Performs close functions unique to indexed sequentially organized data sets.

Stow (CZCOK) -- Stows a member of a partitioned data set and calls the appropriate access dependent close routine.

MOVEPAGE (CZCOC) -- Writes out the POD when required.

RELEXPG (CZCEN) -- Releases unused data set pages when the data set is closed.

DSCB READ/WRIT (CZCEM) -- Read in and write out a format-E DSCB.

ESA LOCK (CZCEJ) -- Lock and unlock the PAT.

DELVAM (CZCFT) -- Deletes a data set, its catalog entry, and the associated JFCB when the "delete at close" option is specified and the DCB being closed is the last one open for the data set.

WRITDSCB (CZCEW) -- Updates the DSCBs on the volume when required.

FREEMAIN (CZCGA) -- Releases virtual storage occupied by the RESTBL, directory pages, and buffer pages.

Release Interlock (CZCOI) -- Releases the interlock on the RESTBL for shared data sets which still have DCBs open against them.

Disconnect (CZCG8) -- Disconnects the task from the shared RESTBL.

SYSER (CEAIS) -- Declares system errors as listed under "Exits".

ABEND (CZACP) -- Abnormally terminates a task under the same error conditions as for SYSER.

Exits:
Normal -- Return to the calling routine.

Error -- SYSER, then ABEND is called under any of the following conditions:

- Search SDST cannot find data set entry.
- Data set being closed is neither sequential, indexed sequential, nor partitioned.

- Error return from Stow.
- MOVEPAGE unable to write out updated POD.
- DELVAM unable to delete a data set and its catalog entry.
- DSCB Read/Writ unable to read or write DSCB.

Operation: For entry at CZCOB1: On entry, CLOSEVAM saves the input registers and establishes base registers for the CSECT, PSECT, JFCB, DCB, RESTBL, DCB header, and member header. If the data set is shared, the RESTBL is interlocked by calling Interlock.

When CLOSE (TYPE = T) is specified for a VAM data set, only data pages, directory pages and DSCBs, where required, are written to external storage. The data set will remain open and in the same condition as would follow a normal OPEN.

Nonpartitioned data sets are positioned (SETL) according to the original OPEN option and data set organization prior to the completion of the CLOSE (TYPE = T). When partitioned data sets are processed, a STOW-R is issued against the checked out members, as during a normal CLOSE. A FIND must be issued by the user if the member is to be reprocessed.

If not a TYPE = T, when the data set is sequential or index sequential, the corresponding access dependent close routine is called. If the data set is partitioned and there is a member still checked out, a call is made to STOW to update the POD and to close the member by calling the appropriate access dependent close routine. If an error return is received from STOW, SYSER and ABEND are called.

Following the above processing or if the data set is partitioned but has no members checked out, a test is conducted to determine if the POD must be written out. Four conditions must exist for this to occur:

1. The data set must be partitioned.
2. There must be directory pages to output.
3. Either the DSCB integrity bit or the POD integrity bit must be set.
4. The DCB being closed must not represent the secondary copy of a duplexed data set.

If all these conditions are met, a call is made to MOVEPAGE to output the directory. During the testing for these conditions, a further error test is conducted. If the data set does not prove to be sequential, indexed sequential, or partitioned, SYSER and ABEND are called. This same error exit is taken if MOVEPAGE is not successful.

Closing the Last Open DCB: Following a successful call to MOVEPAGE or if no call is required, it is determined if this is the last DCB open for the data set. For nonshared data sets this is indicated by a zero count of open DCBs in the JFCB. For shared data sets this same count must be zero and, also, the count of open DCBs for all tasks must be zero. If one of these conditions is met (that is, this is the last open DCB for the data set) it may be necessary to delete the data set or to free unused space. If there are available unused pages and the HOLD option has not been specified, a call is made to RELEXPG to release the pages. The parameter list to RELEXPG includes the RESTBL external page entries representing the unused pages. For a shared data set the interlock words must be removed from the RESTBL before the parameter list can be built.

Following the call to RELEXPG or if no release is performed, the DSCB integrity bit is checked to see if the DSCBs must be updated. If so, a call is made to WRITDSCB to perform the update. After a successful call to WRITDSCB or if the DSCBs were not updated, buffer space area may be freed as described below.

Final Close Processing: In this case, the closing of the last DCB for a data set, the final close processing involves the release of any buffer or overflow pages and the release of directory or POD pages. These releases are accomplished by calls to FREEMAIN. FREEMAIN is also called to release any existing RESTBL pages. Lastly, the RESTBL pointer is cleared from the JFCB, the DCB and DCB header are unchained, the macro transfer list is cleared from the DCB, and then a check of last close for a DELVAM call is performed. If the Delete at Close flag is set, a call is made to DELVAM to delete the data set, its catalog entry, and the associated JFCB. If DELVAM returns an error code, SYSER and ABEND are called.

If the Delete flag is not set or delete has been performed, the control will now return to the caller.

Other Open DCBs: When other DCBs are open for the data set, either for this task or for another task sharing the data set, the data set cannot be deleted nor can unused data set pages be released.

If the data set is partitioned and linked to a member, the number of users

accessing the members is decremented. If this number reaches zero, the member header is deleted from the chain, the chain is updated to account for the deletion, and the deleted header space is added to the list of available RESTBL space. Following this, the DCB header is checked. If it is chained, the header chain is updated to account for the deletion of the DCB header. If the header is not chained, the In Use indicator is cleared. Following either of these actions, the DCB address is removed from the DCB header.

Next, the space occupied by the DCB header is added to the chain of available space in the RESTBL and the DSCB integrity bit is tested. If this bit is set, WRITDSCB is called to update the DSCBs on the volume. After this the final close processing is performed.

For data sets for which there are still open DCBs, the final close processing is abbreviated. Any existing buffer pages or overflow pages are released by calling FREEMAIN and the macro transfer list is cleared in the DCB. Lastly, for shared data sets, DISCONNECT is called to disconnect the task from the shared RESTBL. After this or if the data set was not shared, control is returned to the caller.

For entry at CZCOB2: This entry is from the CLOSE command, and is provided to perform cleanup of virtual memory associated with a data set and occurs when an unavailable (already unloaded or erroneous) DCB cannot itself be closed. The address of the DCB header is pointed to in register 1 on entering CLOSEVAM. CZCOB2 decrements the TDTOPN count and normal processing follows -- but at no time will a DCB be pointed to, and no attempt will be made to perform a STOW, output buffers or directories, or rewrite DSCBs.

DUPCLOSE Routine (CZCEZ)

The function of DUPCLOSE is to close the DCBs associated with a duplexed data set by means of a call to Close Common. In addition, DUPCLOSE unlinks the RESTBLs for the data sets and, for the last DCB open for the data set, unlinks the TDTs and clears the duplicate copy indicator. (See Chart MF.)

Attributes: Privileged, fetch protected, closed, read only, reentrant, residing in public virtual storage.

Entry Point: CZCEZ1 -- Entered by means of a type-1 or a type-2 linkage generated by the expansion of the DUPCLOSE macro instruction.

Input: On entry, register one contains the address of a two word parameter list:

Word 1 -- Address of the primary DCB.

Word 2 -- Address of the secondary DCB.

Modules Called:
Close Common (CZCLB) -- A single call is made to close the DCBs and to return the virtual storage they occupied.

SYSER (CEAIS) -- Called by means of the SYSER SVC when one of the DCBs indicates a DSORG other than VAM.

ABEND (CZACP) -- Called to issue a message and terminate the task.

Exits:
Normal -- Return to the calling routine.

Error -- Termination via ABEND macro instruction. SYSER before ABEND if one of the data sets is not of VAM organization.

Operation: DUPCLOSE saves the calling routine's registers and establishes base registers for its CSECT and PSECT. It then retrieves the address of the DCBs, uses them to establish base registers for operations on the DCBs, and places them in the parameter list it will pass to Close Common.

DUPOPEN next tests both DCBs to ensure that the data set organization for each is VAM. If either DCB fails this test, a minor SYSER is declared, the address of an error message is loaded, and ABEND is called. If both DCBs pass the test, the RESTBLs for both data sets are found and base registers are set up. The addresses of both JFCBs are saved for later reference and Close Common is called with the address of the parameter list in register 1.

The close operation is assumed to be successful so no error test is made on return. DUPCLOSE begins searching for the JFCB associated with the primary DCB. It begins its search at the last JFCB and works its way backward through the TDT. As each JFCB is checked, its DD name is compared with the DD name in the DCB until a match is found. If no match is found the procedure is repeated for the secondary JFCB. When the primary JFCB is found, a test is made to see if any other DCBs are opened for the data set. If none is, the duplicate JFCB pointer and the duplicate copy indicator are cleared. Following this or if another DCB is open, the TDT is searched again looking for the secondary DCB. The search is conducted in the same manner, the same tests are performed, and

the same processing occurs as in the case of the primary JFCB.

When both JFCBs have been processed as above, DUPCLOSE returns to the calling routine.

VSAM Close Routine (CZCOQ)

VSAM Close is called by CLOSEVAM (CZCOB) and Stow (CZCOK) to perform terminal processing which is unique to VSAM data sets. (See Chart MG.)

Entry Point: CZCOQ1 -- Entry is by type-1 linkage.

Input: Register 1 contains the address of the DCB.

Modules Called:
FLUSHBUF (CZCOV1) -- Output contents of buffer pages, if necessary.

PUT (CZCOS2) -- Complete the processing of the preceding locate mode PUT.

Exits: Normal return to calling program.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for VSAM Close PSECT and CSECT, DCB, DCB header and RESTBL.

VSAM Close is called to do terminal processing of a data set with respect to a DCB. The last operation field of the DCB (DCBLOF) is tested to determine if work remains to be done due to an outstanding locate mode PUT. If the last operation was a locate mode PUT, the user was given a pointer to the output buffer where the next logical record is placed. It is assumed by the sequential access method that if a locate mode PUT is called, the user has actually placed a record in the space presented. The outstanding PUT must be terminated to include the last logical record in the data set. A call to a secondary entry point (CZCOS2) is done to accomplish that function.

The Last Operation and Hold Last Buffer flags (DCBLOF and DCBHLB) are cleared.

The last record of a data set has logically been included in the data set, that is, data set pointers updated. That record must now be physically included in the data set. If the last record has not been written from virtual storage, a call to FLUSHBUF (CZCOV) is made. An exception to this is undefined format records. Since they occur in page increments, they are written out at PUT time, and VSAM Close need not call FLUSHBUF.

If any discrepancies exist, final housekeeping is performed to adjust the recorded data set length to the physical data set length.

Control is returned to the caller by the RETURN macro.

VISAM Close Routine (CZCQA)

VISAM Close is called by Close and Stow to terminate processing of a data set (member) and output any existing directory pages. (See Chart MH.)

Entry Point: CZCQA1 -- Entry by type-1 linkage (privileged to privileged).

Input: Register 1 contains the address of the DCB to be closed.

Modules Called:
PUT (CZCPA2) -- Complete previous PUT if still active.

MOVEPAGE (CZCOC1) -- Output directory page(s).

Exits: Normal return is made to the caller via the RETURN macro.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for VISAM Close CSECT and PSECT, DCB, DCB header, and RESTBL.

If the last operation was a PUT, PUT is entered at the secondary entry point to complete any function left outstanding from the previous PUT.

If index sequential directories exist, and if the ISD Integrity flag in the RESTBL is set, MOVEPAGE (CZCOC) is called to output them.

Control is returned to the caller by the RETURN macro.

VAM ABEND Interlock Release Routine (CZCQQ)

VAM ABEND Interlock Release is used by VAM to release interlocks that have been set within a task in which ABEND has been invoked. (See Chart MI.)

Entry Points:
CZCQQ1 -- To release only RESTBL interlocks that may be set. Via type-1 linkage.

CZCQQ2 -- To release all interlocks that may be set. Via type-1 linkage.

Input: Register 1 contains the address of a half word field containing the ID of the task involving ABEND.

Modules Called:
Search SDST (CZCQE) -- To update or delete data set or member entries in the shared data set table (SDST).

Interlock (CZCOH) -- To set read or write interlock on a shared data set entry.

Release Interlock (CZCOI) -- To release read or write interlock on a shared data set entry.

Exits:
Normal -- The calling program receives control by use of the RETURN macro, with the T option set and no special return code.

Error -- May ABEND if unable to release the SDST interlocks.

Operation: The normal release process consists of analyzing the DCBHEADER interlock summaries (Figure 24) one at a time, and releasing the appropriate interlocks by calling the existent Search SDST, Interlock, and Release Interlock routines. The DCBHEADER interlock summaries are updated to indicate such release.

Examining DCBHEADERS: VAM ABEND Interlock Release examines the task ID in each DCB header to determine if the data set is used by the task that has invoked ABEND. On finding a DCB header pertinent to the task, the interlock summary is checked and pertinent interlocks are released as stated below. If the comparison between the task ID in the DCB header and that invoking ABEND is unequal, a link is made to the next DCBHEADER within the chain, and the comparison on task ID repeated. When the chain of DCB headers is exhausted (that is, the pointer to the next DCB header is zero), the link to the next JFCB is made and the process repeated until all JFCBs have been examined (that is, the pointer to the next JFCB is zero).

Releasing the Interlocks: The interlock summary in the DCB header is continuously updated as interlocks are set and reset by Interlock and Release Interlock, with the exception of the shared page table number interlock. The DCB header will reflect the condition of this interlock and it will be released as required. The DHDINT field in the DCB header contains the interlock summary shown in Figure 24.

If an interlock is recorded in the DCB header interlock summary as being imposed on the SDST control, all other interlocks within the SDST are released before releasing the SDST control interlock. If no interlock on SDST control byte is recorded in the interlock summary, and if such a lock is found to be in effect, either the interlock was set by the task which invoked ABEND and is recorded in a DCB header interlock summary that has not yet been examined, or the interlock was set by another task. Since it is not known at a given moment which case exists, the first case is assumed, and the DCB headers are scanned for that particular lock. On release of the SDST control lock all other interlocks are released. While the scan is being executed, the SDST control byte is repeatedly checked for release. If release occurs (the second case is indicated) the scan is discontinued and the general release process is executed.

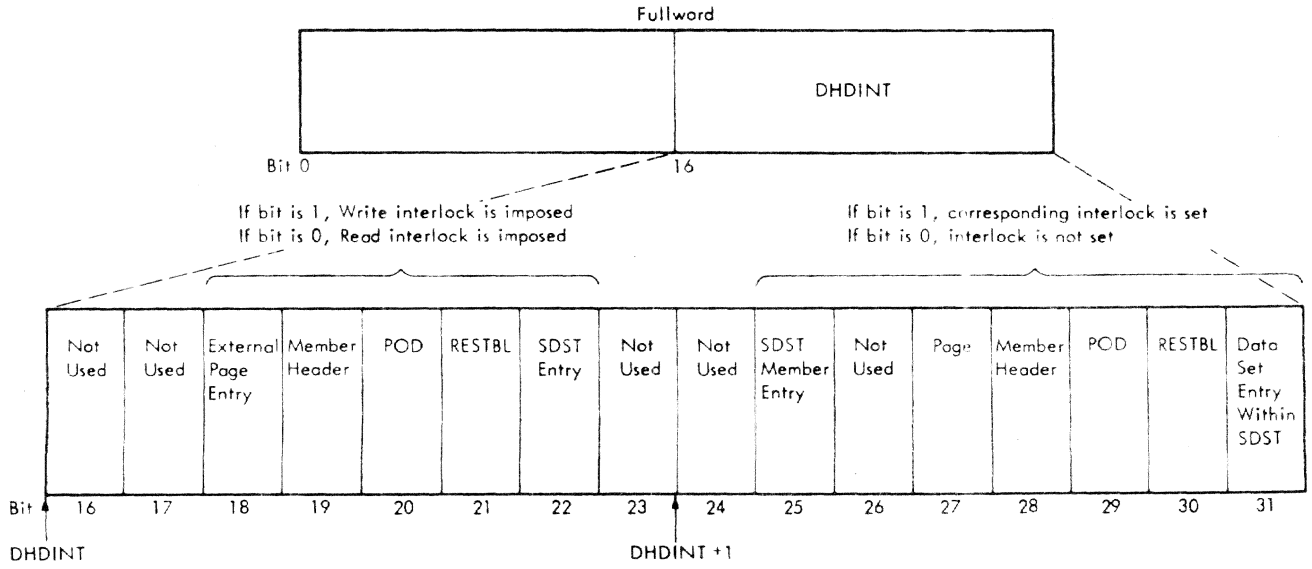


Figure 24. DCBHEADER Interlock Summary

Sequential organization is a subfunction of the VAM user macro instructions. The principal advantage of VSAM is that the maximum record size is one segment (256 pages or 1,048,576 bytes).

The following contribute to VSAM:

User Coded Macro Instructions

DCB	Supply parameters to define data set structure. Also provide work space for the access method modules.
OPEN,CLOSE	Activate/deactivate access to a data set.
GET	Request a record.
PUT	Specify generation of a record.
SETL	Specify access to a particular record or portion of the data set.
PUTX	Update a record in place.

Control Blocks

RESTBL	Relative external storage correspondence table.

VSAM Routines

OPENSEQ	Access dependent initialization to begin processing of a VSAM data set.
CLOSESEQ	Terminate sequential processing.
GET	Sequential access to a record of a VSAM data set.
PUT	Sequential generation of a record and truncation of a data set.
SETL	Specify the record within the data set at which access is required.
PUTX	Update a record without deletion of subsequent records as with PUT.
FLUSHBUF	Process the page buffer contents.

Record formats with the VSAM may be fixed (F), variable (V), or undefined (U) (see Figure 25). These records may span page boundaries, but format-U records must begin and end on page boundaries. To control the processing of format-V records, a length control field is placed in the first four bytes of the record by the user. The access method generates the control data in the first byte and also place a copy of the control field at the end of the record. The pair of control fields (8 bytes) will never span a page boundary.

A description of DCB working storage used by VSAM routines is provided in Table 37.

ROUTINES IN VSAM

VSAM Get Routine (CZCOR)

VSAM Get is called, by expansion of the GET macro instruction, to obtain a data record from a VSAM data set or member. The record obtained may be explicitly identified by a SETL or may simply be the record which sequentially follows that record accessed by the preceding PUTX or GET. (See Chart NA.)

Table 37. Description of DCB Working Storage Used by VSAM Routines

Symbol	Data	Description
DCBNPO	N	Number of pages to output
DCBFPO	N	First page to output
DCBBPU	N	Number of buffer pages in use
DCBHLB	X	Hold Last Buffer flag
DCBLOF	X	Last Operation flag
DCBL01		X'03' SETL
DCBL02		X'0C' PUTX
DCBL03		X'CC' GET - move
DCBL04		X'C3' GET - locate
DCBL05		X'3C' PUT - move
DCBL06		X'33' PUT - locate
DCBPRL	N,X	Previous record length
DCBBP	A	Current buffer position

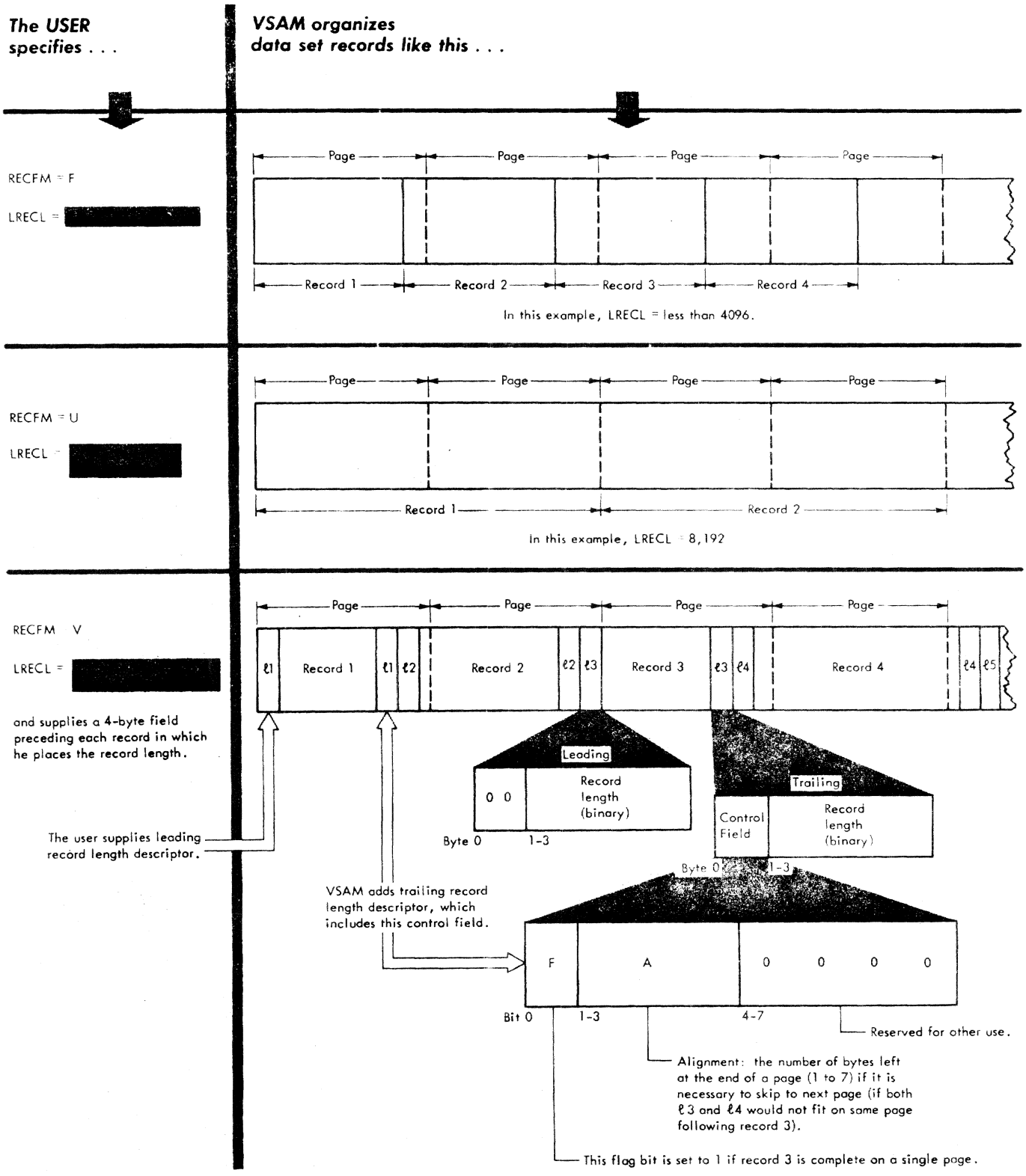


Figure 25. VSAM Data Record and Page Formats

Attributes: Read-only, reentrant, public, nonprivileged, system.

Entry Point: CZCORI -- Get is called by type-1 linkage.

Input: Parameters are passed in general registers as follows:

Register 0 -- If move mode, address of the users area.

Register 1 -- Address of the DCB.

Within the DCB, the macro code field (DCBMCD) has been set to indicate move or locate mode.

Modules Called:
MOVEPAGE (CZCOC1) -- Perform page input.

FLUSHBUF (CZCOV1) -- Purge buffer of data remaining in the buffer since the last operation.

Exits:
Normal -- Exit is made to the user via the RETURN macro.

Error -- ABEND is called under the following conditions:

- The DCB header does not point back to the DCB.
- Record length zero.
- Record length longer than maximum.
- End of data set and missing EODAD.
- Format U record not multiple of a page in length.
- User area not same protection class as the DCB.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for VSAM Get CSECT, DCB, SAVE area, DCB header, RESTBL, and ISA.

VSAM Get is a "fence sitter" and has no PSECT. It is always called by type-1 linkage and assumes the same privilege as the caller. The VSAM Get save area is obtained dynamically by Open Common on a DCB basis. Get has to access the interruption storage area (ISA) to determine its current privilege state before calling privileged routines (MOVEPAGE, ABEND and FLUSHBUF). Get will execute type-1 or type-2 CALLs depending on its privilege state.

The DCB pointer in the DCB header is tested to insure that it points to the DCB being processed. If it does not ABEND is called.

The last operation field (DCBLOF) is tested to see if the last operation was a PUT. If the last operation was a PUT, it means that the data set is positioned at end of data and the current Get would cause an End of Data condition. The End of Data exit in the DCB is taken if the user supplied one; if not ABEND is called.

If the last operation was a Get, a call is made to FLUSHBUF (CZCOV) to clear the I/O buffer before reading in the next record.

The following is common to all other previous operation conditions (SETL, PUTX, Get-move mode Get-locate mode where not more than one buffer page was retained) and to the preceding: If the current record position (page and byte) is beyond the end of data defined in the RESTBL (or member) header, call the user's EODAD routine as described above for PUT.

The retrieval address of the current record and its successor are generated. Following this one of three procedures is executed depending on the record format.

Undefined Record Format (U): ABEND will be executed if the requested record is too large, or if it would not end on a page boundary. The user's EODAD routine will be executed if the end of the record is beyond the end of the data set. In this case ABEND may be executed if no EODAD was specified.

The number of pages to input is computed. If in move mode and the users area begins on a page boundary, MOVEPAGE (CZCOC1) is called to input the record directly into the user area. The last operation is set to GET-move mode. The current relative position is updated and a return is made to the caller.

If in locate mode, or move mode where the user area does not begin on a page boundary, MOVEPAGE (CZCOC1) is called to read the record into the buffer assigned to the DCB. The number of pages in use is set equal to the number of pages input, and the Hold Last Buffer flag is set off. If in locate mode, the buffer address is placed in general register 1, the Last Operation flag is set to GET-locate mode, and a RETURN is made to the caller. If in move mode, the contents of the buffer are moved to the user specified area and the Last Operation flag is set to GET-move mode and a return is made to the caller.

Fixed Format (F): A computation is made to determine if the next record is completely within the buffer. If (CZCOC1) is called to input enough pages to complete the record, starting at the page following the last page in use. The remainder of processing of fixed format records is common to both fixed and variable and is described below, under "Fixed or Variable Format (Common Processing)."

Variable Format (V): If the buffer is empty, MOVEPAGE (CZCOC1) is called to read the next page of the data set into the buffer. The number of buffer pages in use is set to one.

The record length of the record in the buffer is checked to determine if the present contents of the buffer contain the record. If not, MOVEPAGE (CZCOC1) is called to read the additional pages into the buffer at a point following the last page in use. The number of buffer pages is incremented by the number of pages read. The retrieval address of the current record is generated.

Fixed or Variable Format (Common Processing): The record address of the following address is generated. The Hold Last Buffer flag is set on or off depending on whether the current record ends on a page boundary.

If locate mode was requested, the address of the record is placed into general register 1 and a return is made to the caller. If in move mode, the current record is moved to the user area. If more than one page is in use in the buffer, or if the Hold Last Buffer flag is off, FLUSHBUF (CZCOV1) is called to purge the buffer area of the unneeded data.

The Last Operation flag is set to GET-move mode, the current record position is updated and a return is made to the caller.

VSAM PUT Routine (CZCOS)

VSAM PUT is called by the user through the DCB or by CLOSESEQ (CZCOQ), or SETL (CZCOT) to concatenate a record onto the data set and define a new end of data set. When called by the user, the function may be to obtain the address of some buffer space into which the user may construct a record. Any subsequent operation on that data set will result in a call to the secondary entry of this module (CZCOS2) to complete the operation. (See Chart NB.)

Attributes: Read-only, reentrant, public, nonprivileged, system.

Entry Points:
CZCOS1 -- Entered via type-1 linkage. Normal entry point.

CZCOS2 -- Entered via type-1 linkage to terminate an outstanding locate-mode PUT.

CZCOS3 -- Entered via type-2 linkage to permit privileged table storage. PUT enters itself to "jump the fence", become privileged, and update the RESTBL.

Input:
For CZCOS1, register 0 contains the user buffer area (move-mode request). Register 1 contains the address of the DCB.

For CZCOS2, register 1 contains the address of the DCB.

For CZCOS3 -- None.

Modules Called: Calls as follows:
VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function without terminating the task.

FLUSHBUF (CZCOV1) -- Purge data remaining from a previous operation, from the buffer.

Exits:
Normal -- Return to the calling routine.

Error -- VDMEP is called under the following conditions:

- User has READONLY access to data set.
- DCB specified by caller is not addressed by DCB header.
- Record length greater than maximum specified in DCB header (note that this error will not be detected until the following operation is performed when it is caused by a locate mode PUT).
- Undefined (U) format record whose length is not in page multiples.
- Variable format logical record length is too small (less than 4 bytes).

Operation: This module's operation can be summarized as entry; initialization; completing processing of the previous operation; establishing the buffer position for the current operation; if move mode, transferring the data from the user's area to the buffer area; and writing the data on external storage if more than one buffer page is left in use.

At entry CZCOS1, the entry indicator (Q) is set to zero; at entry CZCOS2, Q is set to one. Initialization and general register storage is executed in conformance with

linkage conventions. Base registers are declared for the CSECT, DCB, save area, DCB header, RESTBL and ISA.

VSAM PUT is a "fence sitter", and has no PSECT. It is always called by type-1 linkage and assumes the same privilege as the caller. The VSAM Put save area is obtained dynamically by Open Common on a DCB basis. VSAM PUT has to access the interruption storage area (ISA) to determine its current privilege state before calling privileged routines. It will execute type-1 or type-2 CALLs depending on its privilege state.

Last Operation Not PUT: Data may still exist in the buffer and must be released. If the last operation was a locate mode GET, and more than one page is in use, FLUSHBUF (CZCOV1) is called to release the pages. If the last operation was not a locate mode GET, FLUSHBUF is called if the present position is not at end of data set, and this is not the first PUT. Processing continues at the point, below, "Last Operation PUT Locate Mode."

Last Operation PUT Locate Mode: The record length is obtained from the DCB or from the buffer if the buffer is format-V. If the record length is greater than the maximum, VDMEP is called.

Common Processing to Update Buffer Position: If not format-V, the buffer position is incremented by the record length from the DCB.

If format-V, the current buffer position is updated by the length indication field in the record. A flag is set in the record length control field to indicate that the record is or is not complete within one page. If the number of bytes on this page and following the record is between zero and 7, the control field will be placed on the next page with a gap of between zero and 7 bytes. This gap, the alignment bytes, is accounted for in the record length control fields at the beginning of the record, and in the field between this record and the following record. The control field is then placed following the record and the buffer position is updated by 4.

For either of the above, the Hold Last Buffer flag is set on.

Common Processing to Output Data: The number of buffer pages in use is computed, based on the size of the current record plus the number of bytes in the buffer preceding this record including those in the first page on which it is stored. If the low 12 bits of this number are zeros and this is not a format-V record, the Hold Last Buffer flag off is turned off.

The number of pages which the record occupies is stored as the number of pages to output and also as the number of buffer pages in use. If the Hold Last Buffer flag is off or, if on and more than 1 page is to be output, FLUSHBUF (CZCOV1) is called to empty the buffer. The end of data set parameter in the RESTBL is incremented by the number of bytes in the record just processed.

If the record is format-U the operation is move, and SWITCH is off, the end of data set value is moved from the RESTBL to the DCB. If either the Q or X (move mode) flags are on, the last operation is set to PUT-move mode, and return is made to the caller. This completes processing for the preceding locate mode PUT operation and also certain move mode PUT operations. The processing of the current call is described below.

Processing Current Record: If locate mode, the last operation is set to PUT-locate mode, the current buffer position is placed into general register 1 and, if format-V, the record length of the previous record is placed in the buffer and general register 1 is incremented by 4. Then a return is made to the caller.

If move mode, the X switch is set on. If the record length exceeds the maximum, VDMEP is called.

If records are not format-U or are format-U but the user's area does not begin on a page boundary, the record is moved to the buffer and complete processing with the same procedure described above at "Common Processing to Update Buffer Position" occurs. This also sets the SWITCH field to on for format-U records.

If format-U and the user area is on a page boundary, the Hold Last Buffer flag is turned off, the current buffer position is updated, FLUSHBUF is called to release the pages, and processing is concluded as described above, at "Common Processing to Output Data."

SETL Routine (CZCOT)

SETL is called by VSAM Open (CZCOP), Find (CZCOJ), and by the SETL macro using the V-con and R-cons stored in the DCB to specify access to a particular record within the VSAM data set. (See Chart NC.)

Attributes: Read-only, reentrant, public, nonprivileged, system.

Restrictions: Cannot use SETL B with undefined (U) format records.

Entry Point: CZCOT1 -- This module is called by type-1 linkage.

Input: The following parameters are passed:

Register 0 -- The retrieval address field.

Register 1 -- Address of the DCB.

Retrieval address is specified only when the type code is R. In addition, a type code is preset in the macro field (DCBMCD) to indicate:

B - Beginning of data set.

E - End of data set.

P - Previous record.

R - Retrieval address.

Modules Called:
MOVEPAGE (CZCOC1) -- Input a page into the buffer associated with the DCB.

VSAM Put (CZCOS2) -- Complete preceding locate mode PUT.

FLUSHBUF (CZCOV1) -- Purge buffer of unstored data from preceding operation.

Exits:
Normal -- Return to the calling routine.

Error -- ABEND is called under the following conditions:

- DCB header does not point to DCB.
- Invalid record format in DCB.
- Backspace request for undefined record.
- Attempt to SETL outside limits of data set, with no EODAD exit supplied.
- Request code not defined (not a SETL type code).

Operation: Initialization and general register storage is executed in conformance with linkage standards. Base registers are declared for the CSECT, save area, DCB, DCB header, RESTBL and ISA.

SETL is a "fence sitter" and has no PSECT. It is always called by type-1 linkage and assumes the same privilege as the caller. The SETL save area is obtained dynamically by Open Common on a DCB basis. SETL has to access the interruption storage area (ISA) to determine its current privilege state before calling privileged routines. SETL will execute type-1 or type-2 CALLs depending on its privilege state. If the address of the DCB in the DCB header is not equal to the DCB address passed in the parameter list, or the record format field in the DCB is unspecified, ABEND is used to terminate the task.

If the preceding operation was a locate mode PUT, it will be completed by calling VSAM PUT at entry point CZCOS2.

If the last operation was a locate mode GET and if the Hold Last Buffer flag is on, or more than one buffer page is in use, FLUSHBUF (CZCOV1) is called to release the buffer and, if necessary, return data to external storage. Upon return from FLUSHBUF, the current record position will be updated.

The last operation field is set to indicate SETL and a series of tests is then made to select the proper method of generating a retrieval address in the DCB (DCBLPA).

- Backspace (P): If this is a format-U data set, the task is terminated via ABEND. For variable format records, if the current position is at a page boundary, and this is not the first page of the data set, MOVEPAGE is called (CZCOC1) to read the next page. Then the length control field is obtained for the preceding record and adjust it by the alignment byte count. For fixed format, the record length is subtracted from the current retrieval address (this is done separately on page number and byte position). A negative result causes an exit to the EODAD routine, if one is supplied; otherwise the task is abnormally ended.
- Beginning (B): The retrieval address is set to zero.
- Retrieval address (R): If the retrieval address specified by the caller is beyond the end of the data set, exit is made to the EODAD routine if one is supplied. If there is no EODAD routine an ABEND is issued. If the address is within limits that value is saved in the DCB.
- End of Data Set (E): The end-of-data-set field in the RESTBL (or member) header is moved into the retrieval address field in the DCB.

After the retrieval address has been generated as indicated above, if any buffer pages are in use and the page number of the current page is not equal to that of the retrieval address, the Hold Last Buffer flag is reset and FLUSHBUF (CZCOV1) is called to remove that data from the buffer. If the retrieval address is not beyond the

end of the data set, MOVEPAGE (CZCOC1) is called to read the first page of the specified record into the first page of the buffer. The number of buffer pages in use is set to 1.

The fields in the DCB which define the current record are then set up and a return is made to the caller.

PUTX Routine (CZCOU)

PUTX Rewrite a Logical Record is called by a PUTX macro instruction in the user's program to perform the rewriting of a logical record in a VSAM data set. Initially PUTX checks the validity of the request, and if satisfactory, the buffer page (or pages) containing the record is (are) returned to the data set by linking to the FLUSHBUF (CZCOV) routine. (See Chart ND.)

Attributes: Read-only, reentrant, non-privileged, public, system.

Restrictions: The previous I/O macro instruction for this data set must be a locate mode GET. After manipulating the data, the user may not change the size of the logical record when it is inserted in the buffer.

Entry Point: CZCOU1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB.

Modules Called:
FLUSHBUF (CZCOV1) -- Writes buffer page (or pages) to data set.

VDMEP (CZCQK1) -- Terminates the function and outputs message if user has read-only access.

Exits:
Normal -- Return to the calling routine.

Error -- ABEND is called under the following conditions:

- The DCB header in the RESTBL does not point to the DCB indicated in the PUTX macro instruction.

- The previous I/O operation for this data set was not a locate-mode GET.

VDMEP is called if the read-only access flag is on in the DCB header. flag is on in the DCB header.

Operation: Initialization and general register storage is executed in conformance with linkage convention. Base registers are declared for the CSECT, SAVE area, DCB, DCB header, and ISA.

PUTX is a "fence sitter" and has no PSECT. It is always called by type-1 linkage and assumes the same privilege as the caller. The PUTX save area is obtained dynamically by Open Common on a DCB basis. PUTX accesses the interruption storage area (ISA) to determine its current privilege state before calling a privileged routine. PUTX will execute type-1 or type-2 CALLs depending on its privilege state.

PUTX verifies that the DCB header in the RESTBL points to the same DCB indicated in general register 1, and that the last I/O operation stored in the DCB was a locate mode GET.

If either of the above are not satisfied, an exit to ABEND occurs.

VDMEP is called to terminate the function and issue a message if the read-only access flag is on in the DCB header.

Processing continues by setting, in the DCB, the number of output pages to be rewritten equal to the number of buffer pages in use; the number of the first output page equal to the first buffer page; the indication in the Last Operation flag (DCBLOF) to PUTX.

If there is more than one buffer page to be rewritten, or if a hold does not exist on a single buffer page, FLUSHBUF is called to write out the necessary pages. On return from FLUSHBUF, processing is concluded by updating the current position pointers. If one buffer page is to be rewritten, and a hold exists for this page, FLUSHBUF is bypassed.

Processing concludes by updating the current relative position pointers to reflect the record just processed. This includes the page number pointer, the byte position pointer, and the buffer position pointer. PUTX then issues a normal return.

FLUSHBUF Routine (CZCOV)

FLUSHBUF is called by VSAM Get (CZCOR), VSAM Put (CZCOS), SETL (CZCOT), or PUTX (CZCOU) to return data in the buffer to external storage. (See Chart NE.)

Attributes: Read-only, reentrant, public, privileged.

Entry Point: CZCOV1 -- Entered via type-1 or type-2 linkage.

Input: Register 1 contains the address of the DCB.

Modules Called:
MOVEPAGE (CZCOC1) -- Force buffer contents to be written on external storage.

Insert/Delete Page (CZCOD1) -- Assign external storage to data pages in the buffer.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (without terminating the task).

Exits:
Normal -- Return to the calling routine.

Error -- VDMEP is called under the following conditions:

- Invalid return code from Insert/Delete Page (CZCOD1).
- No external storage space available.
- Storage ration exceeded.
- No secondary storage space specified.
- Attempt to expand RESTBL for shared data set.
- Maximum data set or member size exceeded.
- Insertion beyond end of data set.
- Deletion beyond end of data set.

Operation: FLUSHBUF stores general registers in conformance with linkage conventions and establishes base registers for the DCB, DCB header, and RESTBL or member header.

If the number of pages to output is nonzero, and greater than the number checked out, Insert/Delete Page (CZCOD1) is called to assign enough pages to complete the request. The insertion is made at the page following the last page checked out. The number of pages checked out is increased by the number inserted. Subsequent processing depends on tests made of the Hold Last Buffer flag (HLB), number of pages to output (NPO), and buffer pages in use (BPU), and affects the number of pages checked out as illustrated in Table 38.

After the operation indicated for cases 1, 2, and 3, in Table 38 are performed, if no pages are checked out, the current position is set to indicate the beginning of the buffer, the number of buffer pages in use is reset to zero, and an exit is made to the caller. If pages are checked out and the Hold Last Buffer flag is off, all buffer pages checked out will be released.

The number of pages is reduced by the number of pages that has been checked out. At this time if a page is still checked out, it is moved to the beginning of the buffer area and addresses and counters in the DCB are adjusted to indicate that the current record is in the first buffer page and that one page is in use. This routine then returns to the caller.

Table 38. FLUSHBUF Decisions to Control Buffer Allocations

HLB Flag	NPO	BPU	Action	Resulting NPO	Resulting BPU	Resulting PCO
(1) OFF			Call MOVEPAGE (CZCOC1) to output all pages in the request	0	*	
(2) ON		1			*	
(3) ON	1	1	Same as (1)		*	
(4) ON	1	1	Call MOVEPAGE (CZCOC1) to output all pages except the last. Move last buffer page to the first buffer page and RETURN.	0	1	1

*Terminating action described in text.

VISAM OVERVIEW

VISAM is designed to give the user sequential or nonsequential access to a record within the data (member) by searching for the user specified key. The key is a field located at a uniform position within each record of a data set. The following contribute to VISAM:

User Coded Macros

DCB	Supply parameter to define data set structure.
READ WRITE	Request access or specify creation of records within VISAM data set. Also generates a DECB (parameter list).
SETL	Request location of a specified record.
ESETL	Specify release of a record that was located.
RELEX	Specify release exclusive control of a page.
DELREC	Specify deletion of a VISAM data record.
GET	Request a record.
PUT	Specify generation of a record. Note: Records must be generated in sequence when using this macro.

Control Blocks

DCB	Data control block -- Extended index sequential working storage.
DECB	Data event control block used to control a READ or WRITE operation.
RESTBL	Relative external storage correspondence table.
POD	Partitioned organization directory (if partitioned).

Tables 39 and 40 show fields of the DCB and DECB control blocks used by VISAM routines.

General VISAM Routines

VISAM Put	Sequentially generate or truncate records of a VISAM data set.
VISAM Get	Sequentially obtain records of a VISAM data set.
SETL	Search VISAM data set for a record. Set current key location of a VISAM data set record.
ESETL RELEX	Release a page.
READ WRITE	Nonsequential generation or retrieval of a VISAM data set.
DELREC	Remove a record from a VISAM data set.

Table 39. Description of DCB Working Storage Used by VISAM Routines

Symbol	Data	Description
DCBPCC	N	Data page call counter
DCBOPC	N	Overflow page counter
DCBCL	A	Current VISAM locator position relative to page
DCBCCL	X,A	Contents of current locator X'0000' through X'OFFF' = on-page X'1000' through X'FFFF' = off-page
DCBIOS	X	Input, Output switch
DCBPT	X	Page type X'FF' overflow page X'00' data page
DCBCRL	N	Current record length
DCBCRS	X	Current record switch
DCBRES	X	Read exclusive switch
DCBPLM	X	PUT locate mode only
DCBPMM	X	PUT move mode only
DCBOLM	X	Outstanding locate mode GET
DCBASY	X	Asynchronous switch
DCBRK	A	Record key area for previous record

Add Directory Entry	Add directory entry. Update VISAM key directory.
GETPAGE	Interfaces with VAM General Services to control page I/O.
VISAM Open VISAM Close	Open and close data set (access dependent).

VISAM Page Formats

The following types of page constitute a VISAM data set (member). (Also see Table 41.)

Data and overflow pages consist of data records and locators. The data records are placed on a page starting at the top (lowest numbered bytes) and are added towards the bottom (increasing byte locations). Locators, on the other hand, begin at the end of the page and extend toward beginning (highest numbered bytes to lowest).

The locators are a series of 2-byte fields which, when examined in sequence, give the location of records in ascending sequence by key, regardless of arrangement of records on that page or any number of overflow pages. There are 2 types of locators:

- Onpage locators whose values range from X'000A' to X'0FF8' and specify the starting byte of the corresponding record. The values X'000' through X'009' and X'0FF9' through X'0FFF' are unattainable.
- Offpage locators whose values range from X'1000' to X'FFFF'. The first byte of an offpage locator specifies the page number of the overflow page incremented by 16, on which the data record may be found. The second byte specifies the relative position of a locator on the overflow page which will in turn specify the starting byte location of the proper record.

Keys on directory pages which are write-protected from the user, are automatically placed there as each, except the first, data page is generated.

The lowest key associated with a data page is the one which appears in the directory position corresponding to that page. If a record is added to a page, and this record has a lower key than any other key associated with that data page, the directory is updated to reflect the new value. An inserted record on a page may cause locators to be displaced to subsequent pages -- this case also causes the directory to be updated.

Table 40. Fields and Codes of the DECB Referenced by VISAM Routines (CHADEB)

Symbol	Data	Description		
DECECB	X	Operation completion code.		
DECEC0		X'00' Read/Write code.		
DECTYP	X	Operation Type Code.		
DECTY1	X	First byte of operation field defined as:		
		Code		Operation
		Internal	External	
DEC40		X'40'	KR	Write replace by retrieval address.
DEC43		X'43'	KS	Write replace by key.
DEC44		X'44'	KT	Write new key.
DEC48		X'48'	KY	Read by key.
DEC49		X'49'	KZ	Read by retrieval address.
DEC4A		X'4A'	KX	Exclusive read by key.
DECTY2	X	Second byte of operation field is defined as operation modifier.		
DECT9M		X'40' Character "S" appeared in the area field of the macro instruction.		
DECLEN	N	Data area length.		
DECDCB	A	Location of DCB.		
DECKAD	A	Location of the key or retrieval address.		

Table 41. Organization of a VISAM Data Set

Title	Optional	Max. Pages	Description
Directory (D)	Yes	255	Consists of the lowest key from each data page after the first. Appears in data sets (members) consisting of more than 1 data page.
Overflow (O)	Yes	240	Contains data records and locators when there is not sufficient space on the data page with which its key is associated.
Data	No	65,000	Contains data records and sequentially organized locators.

In order to reduce paging on VISAM data sets with large directories, a super indexed sequential directory (SISD) is created when the number of directory pages exceeds 2 and a SETL by KEY is performed. The key fields in the SISD consist of the first whole key on a directory page, and the address of that key relative to the beginning of the directory page on which it resides (byte 0 = directory page number; byte 1 = key offset within page). The format for the SISD is shown in Table 42.

Table 42. VISAM Page Formats -- Super Indexed Sequential Directory

Bytes	Symbol	Contents
0-1	PAGNUM	No. SISD pages.
2-3	VMSPACE	No. VM pages occupied by the directory.
4-5	ENDSISD	End of SISD directory relative to the beginning of the last SISD directory page.
6-7	ENDDIR	End of ISD directory relative to beginning of last ISD directory page.
8-9		X'0000'
10-4095		Keys and ISD relative addresses. Note that keys and its address may span pages.
When space is allocated for the SISD and until it is updated, bytes 4-7 will contain X'FFFFFFFF'.		

A READ or SETL by key will cause the following:

- Search directory for the largest key which is smaller than or equal to the one supplied by the user.
- Access the indicated data page.
- Using the locators on that page, search for the desired record. Note that this may cause overflow records to be read.

The relationship between locators and records, data pages and overflow pages is depicted in Figure 26. This diagram illustrates variable length records, overflow records, available space and recoverable space. To summarize, the sequence of records is determined not by their sequence on a page, but by the order of the locators which address the records.

The formats of data and overflow pages are shown in Table 43. The format of directory pages is shown in Table 44.

VISAM ROUTINES

The VISAM processing for VISAM Put, VISAM Get, SETL, READ/WRITE DELREC, GETPAGE, and Add Directory Entry is described below.

Table 43. VISAM Page Formats -- Data or Overflow

Bytes	Symbol	Contents
0-1	PAGNUM	Page number (starts from zero)
2-3	ENDDAT	End of data relative to PTR to the beginning of unused spaces
4-5	DATSPA	Data space. Total number of bytes containing data records
6-7	ENDDIR	End of space accounting area (bytes 0-9) in this page value is X'0009'
8-9	RECSPA	Recoverable space. Number of bytes deleted from the data area
10-ENDDAT		Data records and recoverable space
(ENDDAT+1) to (EOS-1)		Unused space
EOS-4093		Locators (halfword each)
4094-4095	EOS	End of available space relative to start of page. Points to start of locators

Table 44. VISAM Page Formats -- Directory

Bytes	Symbol	Contents
0-1	PAGNUM	Total number of directory pages (starts from one).
2-3	DIRPG	Number of pages in directory, used for maintenance of the virtual memory length of the directory buffer after the directory is read into the buffer.
4-5		X'0000'
6-7	ENDDIR	End of directory relative to pointer to the beginning of last data page.
8-9		X'0000'
10-4095		Keys. Note that keys may span pages.

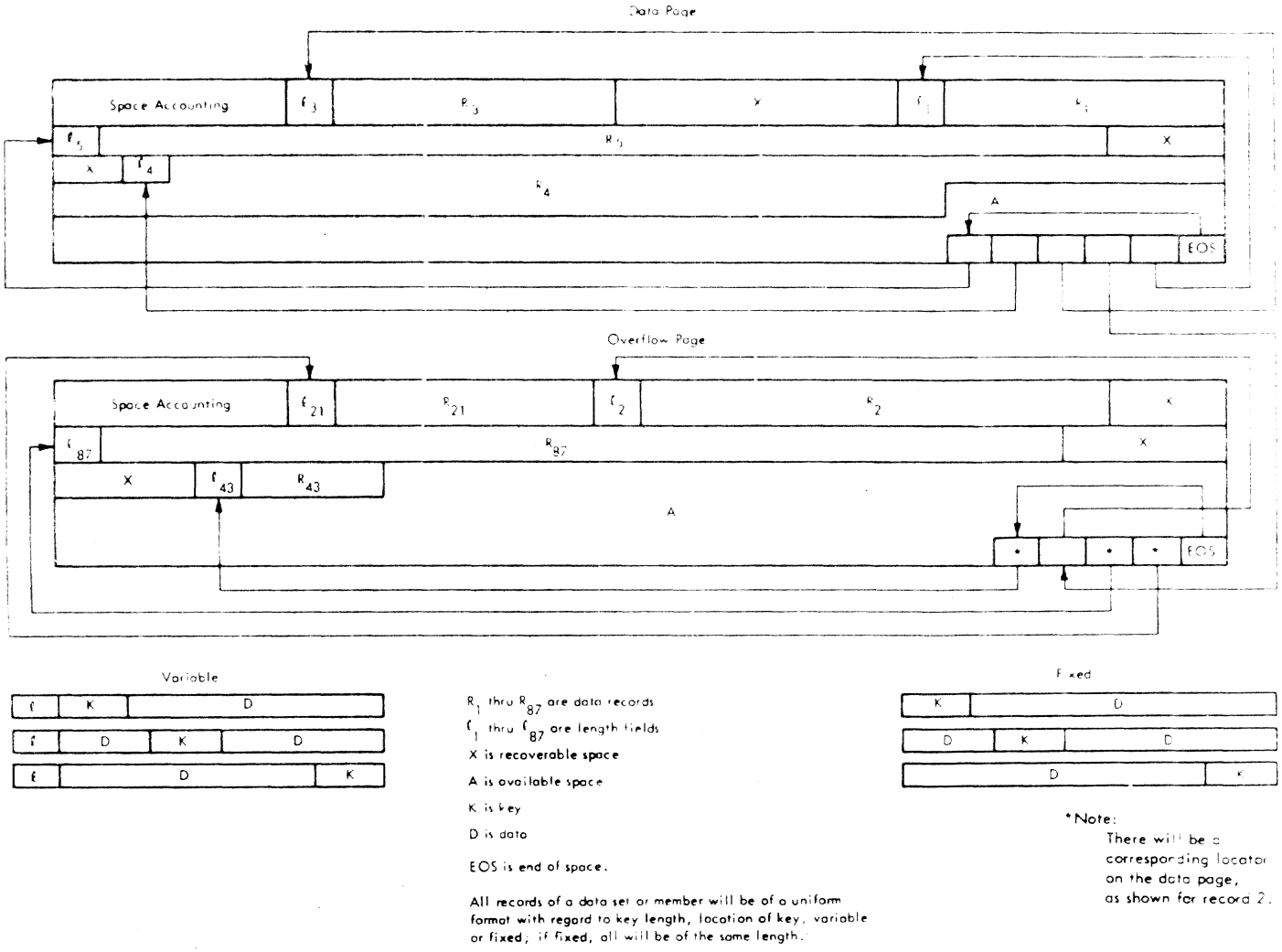


Figure 26. VISAM Record Relationship

VISAM Put Routine (CZCPA)

VISAM Put is called by the user (through the DCB) to concatenate a record in the user's area onto the data set, or to provide the user with a buffer area into which to construct a record. This module may also be called by GET (CZCPB), GETPAGE (CZCPI), SETL (CZCPC), READ/WRITE/DELREC (CZCPE), or VISAM Close (CZCQA) to complete processing of a preceding locate mode PUT. (See Chart OA.)

Attributes: Read-only, reenterable, privileged, public.

Entry Points:
CZCPA1 -- Normal entry from expansion of PUT macro instruction. Entered via type-1 linkage.

CZCPA2 -- Entered to complete previous locate-mode PUT.

Input:
For CZCPA1, register 0 contains the address of user area (move-mode only). Register 1 contains the address of the DCB. The macro code field of the DCB is set as follows:

'0000' Locate mode.

'0004' Move mode.

'0008' Move mode complete.

For CZCPA2 -- Register 1 contains the address of the DCB.

Modules Called:
GETPAGE (CZCPI) -- Perform input and/or output of pages.

Add Directory Entry (CZCPL1) -- Update the index sequential directory.

VDMEP (CZCQK1) -- Terminate the function if read-only access is indicated.

Exits:

Normal -- Control is passed back to the caller by use of the RETURN macro. General register 1 will always contain the address of the record.

Error -- Depending on the error condition, an exit may be made via ABEND or VDMER or, where provided, to a user's SYNAD routine.

ABEND occurs if:

- The record length exceeds 4000 bytes.
- The record length exceeds the maximum stated in the DCB (when completing a previous PUT).
- A SYNAD condition is encountered during completion of a final PUT.

VDMER occurs if:

- The user has read-only access.
- The DCB is not open for output.
- Two or more DCBs are open with OUTPUT specified for a nonshared data set.
- An unexpected return code is received from GETPAGE.

Conditions for which the user may provide a SYNAD routine and the codes which PUT will provide in DCBEX2 are:

Keys equal (sequence error)	X'04'
Keys out of sequence	'0C'
Record length exceeds that specified in DCB	'1C'

Operation: VISAM Put is a "fence sitter," and has no PSECT. It is always called by type-1 linkage, and assumes the same privilege as the caller. The Put save area is obtained dynamically by Open Common on a DCB basis. Put has to access the Interruption storage area (ISA) to determine its current privilege state before calling privileged routines. Put will execute type-1 or type-2 calls depending on its privilege state.

PUT Normal Entry: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the CSECT, DCB, DCB header, RESTBL and data page. Initially, VDMEP is called via the VDMER macro instruction if the user has read-only access to the data set, if the data set is not open for output, or if two or more DCBs have been opened for the data set with the OUTPUT option.

If the previous operation was a PUT, an internal subroutine, "Complete Previous PUT" is executed. Processing continues to find space for a new record.

If any pages exist beyond this page, the number is computed and GETPAGE (CZCPI3) is called to delete them. If this is not the first data page, Add Directory Entry (CZCPL1) is called to update the directory.

If any records exist on this page beyond the current position, they are deleted one by one until the locator indicated by the end-of-space field is reached. Each record deleted causes the data space to be increased by 2 bytes, and recoverable space to be incremented by the length of the record.

Find Space for New Record: The last operation switch is set to PUT. If a page is assigned, and either more than 4000 bytes are available, or padding space, plus a locator, plus record length is not greater than data space, then a page is not required. If a page is not assigned or insufficient space exists, GETPAGE (CZCPI1) is called to insert a new page. The current locator address is set to 4092, the new page switch is set on and the addresses of the current locator and buffer position are generated.

If move mode, the record is moved from the user area to the data page after checking that the record length exceeds neither 4000 (ABEND), nor the value stated in the DCB as a maximum (SYNAD).

The current buffer address is placed in register 1. If the operation being performed is not a PUT-move mode complete, control is returned to the caller. If the operation was PUT-move mode complete, the previous PUT is completed. Control then is returned to the caller.

PUT Secondary Entry: Base registers are established as above, SKIP is turned on and the internal subroutine "Complete Previous PUT" is executed. GETPAGE (CZCPI3) is called to output the data page. A RETURN is executed.

Complete Previous PUT: The key in the current record is compared against the key of the previous record. If out of sequence or equal, SYNAD exits are executed. If this is a new page, Add Directory Entry (CZCPL1) is executed to update the directory. If format-V, ABEND is executed if user exceeded the record length limit in the DCB.

End data and data space are updated by the length of the current record. End of space is updated and a new locator is inserted. The subroutine returns to the portion of this module that called it.

VISAM Get Routine (CZCPB)

VISAM Get is a routine called by the user through the DCB to obtain access to the next logical record. (See Chart OB.)

Attributes: Read-only, reenterable, public, privileged.

Entry Point: CZCPB1 -- Expansion of the GET macro generates linkage to this module.

Input: Parameters are passed as follows:

Register 0 -- Address of user area.

Register 1 -- Address of DCB.

The macro code in the DCB (DCBMCD) is set to indicate move or locate mode.

X'0000' = Locate mode.

X'0004' = Move mode.

Modules Called:
PUT (CZCPA2) -- Complete outstanding PUT operation.

GETPAGE (CZCPI2) -- Input next page of the data set.

Exits:
Normal -- General register 1 will contain the address of the record.

EODAD -- Position is at end of data set.

Error -- ABEND EODAD is unresolved.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for DCB, GET save area, CSECT, and ISA. Base registers for DCB header, data page, and RESTBL are declared as needed.

VISAM Get is a "fence sitter," and has no PSECT. It is always called by type-1 linkage and assumes the same privilege as the caller. The Get save area is obtained dynamically by Open Common on a DCB basis. GET has to access the interruption storage area (ISA) to determine its current privilege state before calling privileged routines. GET will execute type-1 or type-2 calls depending on its privileged state.

The DCB (DCBPLM) is tested to determine if the last operation was a PUT. The secondary entry point of PUT (CZCPA2) is entered to terminate any outstanding PUTs.

Before getting the record, if the data set is not shared and the data page in the buffer is not the current data page, the buffer data page is output via GETPAGE prior to bringing the current page into the buffer.

The Current Record switch in the DCB (DCBCRS) is tested to see if the DCB is currently positioned to the proper record. If the Current Record switch is on, the current locator is obtained. The locator may be a data page locator or point to an overflow page. If it is an overflow page pointer the overflow page is obtained by a call to GETPAGE (CZCPI). The next logical record is located on the data page or the overflow page. The record address, key address, and retrieval address are set from the found record.

If a move mode GET was requested, the record is moved to the user supplied area. Control is then returned to the caller by the RETURN macro.

If the Current Record switch was not on, the locator of the next logical record must be obtained. If the current locator is not at the end of the locators on the current page, the location of the locator is updated to the next logical record and processing continues as if the Current Record switch were on.

If the current locator was the last one on that page, the page number is incremented and the next page obtained by calling GETPAGE (CZCPI). If the call to GETPAGE caused an end-of-file condition, the user-supplied (in the DCB) EODAD exit is taken, if it exists. If the user did not provide an EODAD, VDMEP is called. For a non-end of data condition, the locator pointers are initialized and processing continues as if the Current Data switch were on.

SETL Routine (CZCPC)

SET LOCATION (SETL) is called by the user or by READ/WRITE to locate a specific record within a VISAM data set or member. (See Chart OC.)

Attributes: Read-only, reenterable, public, privileged.

Restrictions: May not SETL by retrieval address for a shared data set.

Entry Points:
CZCPC1 -- Entered via type-1 linkage through expansion of the SETL macro instruction.

CZCPC2 -- Entered via type-1 linkage from other VISAM routines.

Input:
For CZCPC1, register 0 contains the address of key or retrieval address. Register 1 contains the address of the DCB. The macro code field of the DCB will be set to one of the following:

'00' SETL operation to beginning.

'08' By key.

'0C' By retrieval address.

'14' To previous.

'18' To next.

'80' To end.

For CZCPC2, register 1 contains the address of a two-word parameter list:

Word 1 -- Address of DCB.

Word 2 -- Address of key or retrieval address.

The macro code field of the DCB has the same permissible values as for entry point CZCPC1.

Modules Called:
PUT (CZCPA2) -- Complete previous PUT.

GETPAGE (CZCPI2) -- Read data or overflow page without outputting old page.

Interlock (CZCOH1) -- Lock RESTBL for shared data set.

Release Interlock (CZCOI1) -- Unlock RESTBL.

Exits:
Normal -- Return is to the calling program via RETURN macro and with a completion code in general register 15.

Error --

SYNAD: Error return to the user is by using SYNAD exit in the DCB for any of the following conditions:

- Invalid retrieval address.
- Operation was SETL to next, and at end of data set (member).
- Operation was SETL to previous, and at beginning of data set (member).
- Key not found.

ABEND is executed after the following cases where continuation is not meaningful:

- Invalid code in macro code field of DCB.
- Attempted SETL by retrieval address for shared data set.

VDMEP is called if:

- SYNAD address in DCB had not been resolved.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the SETL save area and CSECT, DCB, DCB header, RESTBL, and data page buffer.

SETL is a "fence sitter," and has no PSECT. It is always called by type-1 linkage and assumes the same privilege as the caller. The SETL save area is obtained dynamically by Open Common on a DCB basis. SETL has to access the interruption storage area (ISA) to determine its current privilege state before calling privileged routines. SETL will execute type-1 or type-2 calls depending on its privilege state. The RESTBL is interlocked for shared data sets. The lock will be released prior to any return.

If a PUT was in progress, it is completed by calling PUT (CZCPA2). For SETL to next, the Current Record switch is turned on and the page is read into the buffer. The current locator is stepped down by 2 bytes. If the current locator is equal to end of space (EOS), the next page must be read. The current locator is then set to X'FFC' (4092). This process is repeated until a data record is found.

For SETL to previous, the current locator is stepped up by 2. If this value reaches X'FFC' (4092) the previous page must be obtained by calling GETPAGE (CZCPI2). The current locator is then set to the value in EOS. This process repeats until a data record is found.

Set SETL to beginning, the first data page is read in, the current locator is set to X'FFC' (4092) and a procedure as in "SETL to next" is followed.

For each of the above, the locator is examined; if necessary the specified overflow page will be read into the overflow buffer, and the locator (on page or from the overflow page) is used to set up the retrieval address in the DCB.

For SETL by retrieval address, the specified record is read in and the program exits.

For SETL by key, a binary search of the directory is performed. If a directory is not present, the first page is assumed to contain the specified record. When the required page is read in, a binary search against the keys associated with that page is performed using the locators found there. Overflow pages may be read to perform this search. When the desired record is found, the retrieval address is set as above and the program exits to the caller.

Read/Write, DELREC Routine (CZCPE)

Read/Write is used for nonsequential access to a VISAM data set by either key or by retrieval address. DELREC is used to delete a record by the same criteria. (See Chart OD.)

Attributes: Read-only, reenterable, public, privileged.

Restrictions: Cannot expand RESTBL for shared data set. Shared data sets cannot be read by retrieval address.

Entry Points:
CZCPE1 -- Entered via type-1 or type-2 linkage through expansion of the READ or WRITE macro instruction.

CZCPH1 -- Entered via type-1 or type-2 linkage through expansion of the DELREC macro instruction.

Input:
For CZCPE1, register 1 contains the address of the DECB. The operation will be controlled by specifications stored in the DECB (Table 40).

For CZCPH1, register 1 contains the address of the DCB.

Modules Called:
PUT (CZCPA2) -- Complete previous PUT.

SETL (CZCPC2) -- Locate proper record position.

Add Directory Entry (CZCPL1) -- Update VISAM directory.

VMA (CZCGA) -- Get (free) a page buffer in order to reclaim needed space from a data overflow page.

GETPAGE (CZCPI) -- Input or output of a page.

MOVEPAGE (CZCOC) -- Release read interlock on data page.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

Exits:
Normal -- Return to the calling routine.

Error --

1. Return to the calling routine with a return code of X'04' in general register 15, followed by a call to the user SYNAD routine, under any of the following conditions:
   - READ, WRITE, or REPLACE key not found
   - Equal keys found on a write new key
   - Too many overflow pages
   - Attempt to EXPAND shared data set.
2. VDMEP is called if:
   - No SYNAD address.
   - Update with read-only access.
   - Nonprivileged caller/data set privileged.
   - Input key does not match the key in the record for a write operation.
   - Unexpected return code from CZCPI.
3. ABEND is issued if the DCB header does not point to the DCB, or if the record length exceeds maximum.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the CSECT and PSECT, DECB, DCB, DCB header, RESTBL and data page buffer.

Upon entry to CZCPE1 or CZCPH1, the SYNAD indicator in the DCB (DCBEX1) is set to either READ/WRITE or DELREC. The RESTBL is interlocked for shared data sets. The lock will be released prior to any return. If the last operation was a PUT, a test is made to see if the data set is shared. If shared, PUT is called at CZCPA2 to complete the previous PUT. If the data set is not shared, and the operation is not WRITE New Key, PUT is called at CZCPA2. If WRITE New Key and the key is not greater than the previous key, CZCPA2 is called. However, if the new key is greater than the previous key, PUT is called at CZCPA1 to place the new record with a move mode PUT.

The type of the requested operation (by key or by retrieval address) is determined,

and the appropriate code is set in the DCB macro code field (DCBMCD) prior to calling SETL. A check for read-only access is made; if so, VDMEP is called. SETL (CZCPC2) is then called to locate the desired record.

SETL returns either found or not found. If not found, the key did not exist and the operation is assumed to be a WRITE New Key. At this point for operations other than WRITE New Key, SYNAD is called with a not found condition. For a new WRITE, not positioned to the end of the data set, a locator is inserted in the appropriate place, the control information governing space is updated to reflect the insertion, and the data record is moved. GETPAGE (CZCPI) is called to output the page containing the new record, and control is returned to the caller by the RETURN macro. If the operation is WRITE New Key, positioned to end of data set, and shared, PUT is called at CZCPA1 with the PUT option set to Move Mode Complete (DCBMCD = X'0008'). If the data set is not shared, the PUT option is set to Move Mode (DCBMC1 = X'0004'). Upon return from PUT, control is returned directly to the caller if the data set is not shared; if shared, GETPAGE (CZCPI1) is called to output the page containing the new record prior to returning to the caller.

If SETL returned a found condition, the operation to be performed is one of the following:

- DELREC
- READ
- WRITE Replace by Retrieval Address
- WRITE Replace by Key

DELREC Operation: If the operation is a Delete, the locator is removed from the page and the remaining locators are compressed to close the gap made by the deletion of the locator. The number of bytes occupied by the deleted record and its locator is added to the recoverable space counter and the End of Space pointer is updated to reflect the deletion. If the physical deletion was performed on an overflow page, the overflow page is written out by calling GETPAGE (CZCPI1), in addition to calling GETPAGE to output the data page. The locator on the data page is cleared prior to outputting the page. Control is returned to the caller by the RETURN macro.

READ Operation: For the READ operation, the data record located by SETL is moved to the user specified area. A test is then made to determine if the operation being performed is a READ Exclusive (READ KX). If yes, control is returned to the caller. If not, and the data set is not shared, control is returned to the caller. If shared, MOVEPAGE (CZCOC) is called to release the READ interlock on the data page. Control is then returned to the caller.

WRITE Replace by Retrieval Address and WRITE Replace by Key Operation: For a WRITE Replace, where record length is equal to or shorter than the old record, the new data replaces the old, recoverable space is adjusted, and control is returned to the caller. For a WRITE Replace where the new record is longer than the existing record, the procedure for a Delete is followed to adjust available space. Then the record will be moved into the available space if sufficiently large. If space plus recoverable space is not sufficient to contain the record, the record will be placed on an overflow page where sufficient space exists.

When space plus recoverable space is sufficient to contain the record, GETMAIN (CZCG2) is called to obtain a 1 page buffer and the page (overflow or data) will be copied so as to collect available space into a single field, thus allowing the new record to be inserted. FREEMAIN (CZCG3) is then called to release the page buffer. Alternatively, a WRITE may cause a page to become filled with locators or cause an existing record to be moved to an overflow page.

Add Directory Entry will be called to update the directory if the last locator on the current page must be moved to the next page. This process may repeat when pages filled with overflow locators are encountered.

GETPAGE Routine (CZCPI)

GETPAGE is used by VISAM routines READ/WRITE (CZCPE), SETL (CZCPC), VISAM Get (CZCPB), and VISAM Put (CZCPA), to control page I/O of a data set or member. It is used to bring data or overflow pages of a data set into a buffer (or locate them), delete existing pages, add new pages, or update existing pages. GETPAGE calls MOVEPAGE to set up the action I/O operation. Additionally, GETPAGE releases interlocks on the external page entries (CHAEPE) of a shared RESTBL, RELEX and ESETL. (See Chart OE.)

Attributes: Read-only, reenterable, public, privileged.

Entry Points:

CZCPI1 -- Output current page, read specified page; if page number equals last data or overflow, insert one page.

CZCPI2 -- Input a specified page.

CZCPI3 -- Delete specified pages.

CZCPG1 -- Release exclusive control WRITE interlock of a page (RELEX).

CZCPD1 -- Release existing READ interlock on a page (ESETL).

Type-1 or type-2 linkage may be used for any of the above entry points.

Input: Register 1 contains the address of the DCB.

Modules Called:
Stow (CZCOK1) -- Update POD for overflow page in VPAM member.

MOVEPAGE (CZCOC1) -- Input or output a page.

Insert/Delete Page (CZCOD1) -- Insert added pages or remove unused pages. (CZCOD2) -- Delete page from a data set or member.

Interlock (CZCOH1) -- Called by CALL macro to lock RESTBL.

Release Interlock (CZCOI1) -- Called by CALL macro to release RESTBL lock.

WRITEDSCB (CZCEW1) -- Update DSCB for a new VISAM overflow page.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

VISAM PUT (CZCPA2) -- Complete PUT if that was last previous operation.

Exits:
Normal -- Return to the calling routine with one of the following return codes:

'00'	Normal.
'04'	Insert required, DCBIO switch not set for input.
'0C'	Maximum overflow pages exceeded.
'10'	Return code of '04' received from MOVEPAGE (CZCOC1).
'14'	First attempt to assign overflow page with no external space available.

Error -- VDMEP is called under any of the following conditions:

- Incorrect page number on attempt to insert page.
- No external space available.
- Storage ration exceeded.
- No secondary storage allocation.
- Attempt to expand shared data set.
- Maximum data set or member size exceeded.
- Attempt to insert or delete beyond end of data set.
- SYNAD, but no SYNAD address in DCB.
- Invalid return code from CZCOD (Insert/Delete Page).

Operation: The several entry points serve processing as follows:

CZCPI1 and CZCPI2: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the GETPAGE CSECT and PSECT, DCB, DCB header and RESTBL.

Entry flags are set to indicate which entry point was called. CZCPI1 effects the outputting of the current page before performing the input of the requested page. CZCPI2 effects the inputting only.

GETPAGE may be entered to insert a new page in the data set. Insert/Delete Page (CZCOD1) is called to insert the page. WRITEDSCB will be called to update the DSCB if a new overflow page is being inserted. The page is initialized to VISAM format and control is returned to the caller by the RETURN macro.

For a new insert which has a page currently in the buffer, a call is made to MOVEPAGE to output the page if GETPAGE was entered at entry point CZCPI1.

Also, if it was a pure output request, control is returned to the caller by the RETURN macro.

For a noninsert entered at entry point CZCPI2, or entered at entry point CZCPI1 for other than output, a test is made to determine if the buffer is empty. If not, and if the data set is shared, MOVEPAGE is called to effect the input of the requested page. If nonshared, and if the page (data or overflow) is already in the buffer, control is returned directly to the caller. If the page is not in the buffer, MOVEPAGE is called to effect the input of the requested page.

CZCPI3: Entry is made at CZCPI3 to delete pages from a VISAM data set. Insert/Delete Page (CZCOD2) is called to perform the physical deletion. The number of data, directory, and overflow pages are updated to reflect the deletion. Control is returned to the caller by the RETURN macro.

CZCPD1 and CZCPG1: Indicators are set to release read or write interlocks. If the last operation was a PUT, PUT (CZCPA2) is called to terminate the previous PUT. The call to PUT causes a call to MOVEPAGE which will release the indicated interlocks. Control then returns to the caller.

If the last operation was not a PUT, control is passed just beyond the CZCPI1 and CZCPI2 entry points. The resulting calls to MOVEPAGE will reset the indicated interlocks.

Interlock Handling: GETPAGE will, upon getting a request to input a page, test if that page is a data page or an overflow page. For an overflow page, it will input the page, since overflow pages are not interlocked. For a data page, GETPAGE will check if there is a data page interlock currently imposed. If so, it will determine if a read or write lock is set, and call MOVEPAGE with the appropriate option to release the lock. It will then input the requested page. If MOVEPAGE returns to GETPAGE with a return code stating that the page was locked and could not be input, GETPAGE will set a return code, and exit, indicating the page was not input.

Add Directory Entry Routine (CZCPL)

Add Directory Entry is used to change contents of a VISAM directory due to added or deleted pages, which may cause the size of the directory to change. It also changes the key value when a change is made to a record that does not cause the number of data set pages to change. (See Chart OF.)

Attributes: Read-only, reenterable, public, privileged.

Restrictions: The directory of a shared data set may not be expanded.

Entry Point: CZCPL1 -- Entry to Add Directory Entry is by type-1 (privileged to privileged), or type-2 (nonprivileged to privileged) linkage.

Input: Register 1 contains the address of the DCB for the member whose directory is to be updated.

Modules Called:
GETMAIN (CZCG2) -- Obtain one page of virtual storage for initial directory entry.

Expand (CZCG4) -- Increase space assigned to the directory area.

Insert/Delete Page (CZCOD1) -- Insert added directory page(s) into RESTBL. (CZCOD2) -- Delete page(s) no longer needed in directory.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

Exits:
Normal -- Return is to calling program using the RETURN macro.

Error -- ABEND is called for the following conditions:

- Current page number greater than number given in DCB.
- Attempted add beyond end of data set.
- Number of directory keys not properly computed.

VDMEP is called (the function is terminated) under any of the following conditions:

- Maximum directory size exceeded.
- No external storage space available.
- Storage ration exceeded.
- No secondary storage allocation specified.
- Expanding directory of shared data set.
- Insertion beyond end of data set.
- Deletion beyond end of data set.
- Maximum data set size exceeded.
- Invalid return code from CZCOD.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the CSECT and PSECT, DCB, RESTBL and DCB header.

If one or no data pages exist, no keys are placed in the directory and control is returned to the caller by the RETURN macro.

If keys are to be taken out of the directory as a result of the deletion of data

pages, the keys are removed; if any directory pages are vacated due to key removal, they are deleted from the data set by calling Insert/Delete Page at CZCOD2. The number of directory pages is updated and control is returned to the caller by the RETURN macro.

If a data key is to be added to the directory, and no directory pages exist, virtual storage space is obtained by calling GETMAIN. The directory page is logically inserted into the data set by calling Insert/Delete Page at CZCOD1. The newly obtained directory page is initialized.

The number of keys in the directory is computed. The number of keys in the directory must be equal to or less than the current data page number minus 1, which means that the call is to change a key. If the current data page number minus 1 is greater than the number of keys, ABEND is called, since an attempt is being made to add data beyond the end of the data set.

If a new key is being added, the last directory page is checked to see if enough room exists to hold the key. If no space is available, and no assigned but unused directory pages exist, the directory is expanded (CZCG4) providing the data set is nonshared. If shared, VDMEP is called. For the nonshared data set, a check is made to determine if more than one DCB has been opened for the data set; if so, all are linked to the same ISD and SISD buffer page. The new directory page is initialized and logically inserted into the data set by calling Insert/Delete Page at CZCOD1. Processing continues as if a new key were being inserted or a key being changed. The key is moved to the directory. If a new key is being added, a new end of directory is computed.

The ISD Integrity flag in the RESTBL is set; this indicates to VSAM CLOSE (CZCQA) that it should write the ISD to external storage.

Control is returned to the caller by the CALL macro.

VPAM OVERVIEW

The partitioned organization of the virtual access method uses both sequential and index sequential organizations to process data set members. This method uses macro instructions and routines to combine VSAM and VISAM data sets into a single data set as a series of logical partitions. To deal with the complexities associated with this access method, TSS/360 uses a partitioned organization directory (POD) and relative page/external page correspondence table (RESTBL). The POD and the associated data control block (DCB) are used to relate a member, or its alias names, to the relative position of the member within the data set. The RESTBL is used to determine the actual external device address of the requested page(s).

The following contribute to the virtual partitioned access method (VPAM).

User Coded Macros

DCB	Supplies parameters to define member structure and attributes.
FIND	Requests access to a member descriptor from the POD.
STOW	Specifies updating of a member and/or alias descriptors in the POD.

Control Blocks

DCB	Data control block - organization-independent working storage.
RESTBL	Relative external storage correspondence table.
POD	Partitioned organization directory.

General VPAM Routines

Find	Locates member descriptor in POD.
Stow	Updates member and/or alias descriptors in POD.
Search	Searches POD for member or alias name.
Extend POD	Increases size of POD.
Relocate Members	Updates the POD to account for in size of the POD or any member of the data set.
GETNUMBR	Gives page number of a member relative to the data set, based on starting page number of the member.

VPAM CONTROL BLOCKS

The POD and RESTBL member header are described below.

Partitioned Organization Directory (POD)

The POD, which consists of page(s) at the beginning of a VPAM data set is placed in the user's virtual storage by OPENVAM at data set OPEN time. The POD page(s) are provided and maintained by VPAM for each partitioned data set. Its function is to document member names with their relative locations within the data set and their attributes (data set organization, number of pages, user data field). The POD also documents all aliases assigned to the various members of the data set.

The maximum size of a partitioned data set is 65,535 pages; the maximum size of all members or even a single member is 65,534 pages. When the data set is shareable, the POD is placed into shared virtual storage and protected from the user.

The directory, Figure 27, is divided into four parts; each POD part provides a particular function. The four parts are as follows:

1. An interlock control entry.
2. A space control entry.
3. A directory block hashing table.
4. Directory block entries linked into 64 chains.

The interlock word and the procedures to update same are described in the module

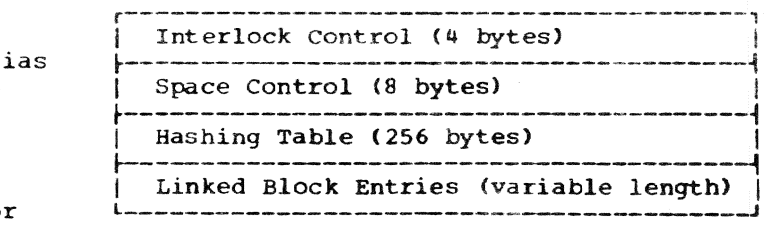


Figure 27. Partitioned Organization Directory (POD)

descriptions for VAM general services routines Interlock (CZCOH), and Release Interlock (CZCOI).

The space control entry (PODSPA) indicates the number of pages in the POD and the number of pages in the data set (Table 45).

The hashing table (PODHT) contains 64 pointers (each four bytes in length to the member and alias descriptors. The hashing value of a member name or alias causes selection of one of the 64 pointers, which in turn contains the byte address relative to the POD, of the linked block entry corresponding to the first name having that hashing value. This block entry then points to the word address relative to the POD, of the second name having that hashing value, etc.

Linked block entries are of two types: member descriptor entry and alias descriptor entry. A member descriptor entry is defined in Table 46.

An alias descriptor is defined in Table 47.

Use of Member Headers in RESTBL

For partitioned data sets only, a field (DHDLNK) in the DCB header (DSECT name: CHADHD, Table 48) of the RESTBL, points to the member header (CHAMHD) rather than to the RESTBL header. The RESTBL member header gives additional data needed to locate pages within a member.

Table 45. POD Format

Bytes	Symbol	Contents
0	PODW	Write interlock
1	PODR	Read interlock
2	PODRC	Read interlock counter
3	PODIC	Control bytes for PODR and PODRC
4-5	PODPG	Number of pages in POD
6-7	PODNDP	Number of pages in data set, including POD pages
8-B	PODLBP	Linked block pointer, pointer (relative to POD) of next available byte in POD
C-10B	PODHT	Hashing Value Table 256 bytes

VPAM ROUTINES

Descriptions of Find, Stow, Search, Extend POD, Relocate Members, and GETNUMBR follow.

Find Routine (CZCOJ)

Find searches the POD in order to locate the member descriptor. The member information including the starting page number and RESTBL header offset is transmitted to the member header and the associated DCB. (See Chart PA.)

Attributes: Read-only, reenterable, privileged, public, system.

Restrictions: A DCB can refer to only one member at a time. STOW must be issued to change user data in the POD prior to issuing a FIND. The DCB must be opened before FIND is executed.

Entry Point: CZCOJ1 -- Via type-1 or type-2 linkage.

Input: On entry, general register 1 contains the address of the following four-word parameter list:

Word 1 -- Address of DCB.

Word 2 -- Address of member name or alias.

Word 3 -- Address of area where user data is to be placed.

Word 4 -- Address of member of bytes of user data.

Modules Called:

Stow (CZCOK1) -- Stow member already checked out to his DCB.

CKCLASS (CKCLS) -- Check protection class of user area and DCB for compatibility.

VDMEP (CZCQK1) -- Process error occurring while trying to access locked member header.

Search (CZCOL1) -- Locate member or alias descriptor in POD.

VSAM Open (CZCOP1) -- Initialize for sequential organization.

VISAM Open (CZCPZ1) -- Initialize for index sequential organization.

Interlock (CZCOH1) -- Impose interlocks on RESTBL for shared data sets.

Release Interlock (CZCOI1) -- Release interlocks on RESTBL for shared data sets.

Table 46. POD Member Descriptor (Part 1 of 2)

Symbol	Data	Field Definition
POMNAM	C	The member name, left-adjusted and padded with blanks, if necessary. For a deleted entry, each byte of this field is set to X'FF'.
POMFLG	X	Bit 0=1 to indicate that this is a member descriptor. Bit 1=0 indicates that no user data is appended. Bit 1=1 indicates that user data is appended. Bits 2-7 are presently not used.
POMHAS	W, X	Bits 0-19 contain a pointer to the next descriptor whose name is a hashing synonum to the member name. This field points to the relative fullword upon which the next hashing synonym descriptor exists, or is zero if no such descriptor exists. Bits 20-21 indicate the organization of the member: 00 Sequential 01 Index Sequential Bits 22-23 indicate the record length format of the member: 00 Variable 01 Fixed 10 Undefined
POMFP	N	First page of this member relative to data set.
POMPG	N	Number of data pages in this member.
POMSEQ	N	Sequential member - record length (actual or maximum)
POMKL	N	Length in bytes of the keys for an index sequential member.
POMIX	N	Bits 0-11 - length of the logical records for a member with fixed length logical records, the maximum logical record length for an index sequential member with variable length records. Bits 12-23 - relative position of the key within a logical record of an index sequential member.

Table 46. POD Member Descriptor (Part 2 of 2)

Symbol	Data	Field Definition
POMOVP	N	Number of overflow pages (index sequential member only).
POMPAD	N	Pad percentage for index sequential pages. Maximum value is 50.
POMDP	N	Number of directory pages, index sequential member, or number of unused bytes in the last data page for a sequential member.
POMBU	N	Number of bytes of user data contained in next field; this field and following field will be absent if bit 1 of POMFLG is 0.
POMUSE	X	Optional data to be supplied by the user. Length of this field is specified by POMBU. For object program modules, this field describes the relative (to the member) locations of the internal symbol dictionary. The next descriptor will begin at a word boundary regardless of the length of this field.

SETL (CZCOT1) (VSAM) -- Position DCB to beginning or end of VSAM member.

SETL (CZCPC2) (VISAM) -- Position DCB to beginning or end of VISAM member.

Expand RESTBL (CZCQI1) -- Expand RESTBL by 1 page to accommodate new member header.

TSEND (CEAP7) -- Function is executed while waiting to access user counter in member header.

CLOSEVAM (CZCOB1) -- Close member header if open on a STOW-type D.

Exits:
Normal -- Return is made to the caller with a completion code in general register 15. If the user area and size were unspecified in the calling parameter list, the location and number of bytes of user data in the member descriptor (POD) are returned as words 3 and 4 of parameter list addressed by general register 1.

Table 47. POD Alias Descriptor

Symbol	Data	Field Definition
POENAM	C	The alias, left-adjusted and padded with blanks if necessary. For a deleted entry, each byte of this field contains X'FF'. Y. For a deleted entry, each byte of this field contains X'FF'.
POEFLG	X	Is zero (high-order bit being zero indicates that this not a member descriptor).
POFHAS	W	Bits 0-19 are a pointer to the next descriptor whose name is a hashing synonym to field 1. This field points to the relative fullword containing the next hashing synonym descriptor, or is zero if no such descriptor exists. Bits 20-23 are zero.
POEMEM	B	Pointer to the member descriptor for which this entry is an alias (byte address relative to the POD).

Return codes:

'00' Member or alias found and member opened.

'04' Member name or alias not found.

'08' The DCB is creating a member -- A STOW-N must be issued for member being created before this DCB can be used to FIND a member.

'0C' Data set organization in member descriptor does not match DSORG specified in DCB. (Can only occur if DCB specified VIP or VSP and member descriptor did not match; if VP was specified in DCB, Find fills DCB in with DSORG found in member descriptor.)

'10' The user area length is not large enough to contain the user data to be retrieved.

'14' Member to be located has already been checked out to this DCB. (Member has been previously found.)

Table 48. RESTBL Member Headers (CHAMHD)

Symbol	Data	Field Description
MHDNAM	C	Member name
MHDFEP	W	Offset to first external page entry of the member
MHDDIR	N	Number of directory pages (index sequential organization only)
MHDDAT	N	Number of data pages in the member
MHDOVF	N	Number of overflow pages (index sequential organization)
MHDBYT	N	Number of bytes used in last page of sequential organization
MHDFLG	X	Flag to indicate sharing status and organization X'80' Shared X'20' Index Sequential
MHDINT	L	Interlock byte to protect number of users and chain link fields
MHDUSE	N	Number of users associated with this member
MHDVAL	X	Value of first external page entry in the RESTBL of the member
MHDNMH	D	Chain to next member header*
MHDPMH	D	Chain to previous member header*

*This value gives the relative location within the RESTBL by multiplying by 8 (left shift 3 bits).

FIND calls VDMEP if:

1. While waiting to access locked member headers, an Attention is received.
2. While waiting to access locked member headers, 100 TSENDs are completed.
3. The DCB is found to be closed or invalid.
4. The DSORG is invalid.

Error -- ABEND will be executed upon detecting any of the following conditions:

- DCB header does not point to DCB.
- RESTBL of a shared data set requires expansion.
- Error return from SRCHSDST call.
- Error return from STOW call.
- The protection classes of the DCB and user area are incompatible.
- DCB Header not linked to member header.
- VPAM member header locked, with no active DCB Header.
- Invalid member or alias name specified.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the Find CSECT and PSECT, DCB, DCB header, RESTBL, member header and POD.

Upon entry, if the DCB header does not point to the DCB, ABEND is called.

The RESTBL is write-interlocked for shared data sets.

The DCB is tested to determine if it is currently in use. If a member had been found previously and was still checked out to the DCB, Find calls Stow (R) to close the member. If the DCB is in use for creating a member, the call to Find may have been done erroneously. A return is given to the caller indicating that the DCB is in use for creating a member. This allows the user to name and stow the member and salvage the work he has done which could have been lost from an erroneous call to Find, since the member being created was unnamed.

If the DCB is not in use, the POD is searched for the name given to Find. If the name cannot be located in the POD, a "not found" return is made to the caller.

If the found name was an alias, the corresponding member name is located.

User data is moved to the user area if it was specified.

Where more than one DCB is open for output for a nonshared member, all DCB headers are linked to the same member header and flags are set to indicate the data set may not be written out.

The active member header chain in the RESTBL is searched to see if the member is already in use by another DCB. If the member is active a test is made to see if it is write-interlocked. This means that the member is currently being modified and is unavailable for use. A time-slice end is requested via TSEND. At the resumption of the user's task, the FIND has to be re-initiated, since the other user may have deleted the member or changed its name while it was interlocked.

If the member was active, the DCB header is linked to the active header. Dummy header space is placed in the deleted header chain if it existed.

If the member was not active, a member header has to be built. It can be built in the dummy header space provided by OPENVAM, in available (deleted header) space, or it can be built in unused RESTBL. A dummy header will exist if this is the first call to Find since OPENVAM was executed. If a FIND was done and the DCB header was linked to an active member header, the dummy header space was released. Subsequent FINDs will use the deleted header space, if available, or unused RESTBL space. If unused RESTBL space is required and there is not enough space remaining to accommodate a member header, the RESTBL is expanded by calling Expand RESTBL (CZCQI) (non-shared data set only).

After the member header is built, data set parameters are filled in the DCB from the POD.

If directory pages exist (VISAM), GETMAIN (CZCGA), is called to obtain virtual storage space and the directories are read in by calling MOVEPAGE (CZCOC).

The data set organization (DCBDSO) is tested. If VSAM or VISAM partitioned, SETL is called, depending on the Open option to logically position the members for processing. If the DSORG is "non-specified" (VP), the data set organization is obtained from the POD and the appropriate access dependent open routine is called.

The RESTBL interlock is released for shared data sets, and control is returned to the caller by the RETURN macro.

Stow Routine (CZCOF)

Stow is used to modify, add, or delete member or alias descriptors in the POD. The RESTBL will also be updated as required. Stow also updates the user data field in the member descriptor. (See Chart PB.)

Attributes: Read-only, reenterable, privileged, public, system.

Restrictions: Each member and alias name within a VPAM data set must be unique.

The DCB used to control operations on a member must be open prior to issuing a STOW.

If a type-N STOW or type-R STOW is given for a member, subsequent references to the same member must be preceded by a FIND. A FIND must be issued before attempting a STOW (R), (U), or (D).

If new aliases are being added to an existing data set, no duplicates are allowed within the input list of new aliases.

Entry Point: CZCOK1 -- Called by either type-1 (privileged to privileged), or type-2 (nonprivileged to privileged) linkage.

Input: Parameters are passed to Stow in general registers as follows:

Register 0 -- Address of the user supplied data area. The formats will be interpreted, depending on the type of STOW issued. They are described with the STOW macro format.

Register 1 -- Address of the DCB. Note that the macro code field in the DCB will be preset to one of the values (hex) discussed with STOW macro format.

Modules Called:

CLOSEVAM (CZCOB1) -- Close member header if open on a STOW type-D.

Search (CZCOL1) -- Search POD for a member or alias name.

Search SDST (CZCQE1) -- Search shared data set table.

Interlock (CZCOH1) -- Impose read or write interlocks on POD and RESTBL.

Release Interlock (CZCOI1) -- Release read or write interlock on POD and RESTBL.

Reclaim (CZCOG1) -- Release external pages.

Relocate Members (CZCON1) -- Updates POD after call to RECLAIM.

Extend POD (CZCOM1) -- Expand POD by page.

VSAM Close (CZCOQ1) -- Close VSAM member.

VISAM Close (CZCQA1) -- Close VISAM member.

GETMAIN (CZCG2) -- Obtain storage for the alias list.

FREEMAIN (CZCG3) -- Release virtual storage.

Expand (CZCG4) -- Obtain additional contiguous storage for the alias list.

Disconnect (CZCGA) -- Logically disconnect a task from an area of virtual storage.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

Find (CZCOJ1) -- Search POD to locate a member descriptor.

Exits:

Normal -- Return with one of the following codes in register 15:

'00'	STOW successful.
'04'	New name, or replacement for old name, already in use (N, NA, C, CA), or old alias does not belong to member specified.
'08'	Member name not in POD.
'10'	Old member name not in POD (C), or alias name not in POD (CA, DA).
'14'	Illegal STOW type requested - Macro code out of range. - Name specified is all FFs. - Input area not on FW boundary. - STOW (NA) and alias count=0.
'18'	User data exceeds maximum length.
'20'	Attempt to expand POD for system catalog.

Error -- When any of the following errors are detected, the ABEND procedure is executed to terminate the task:

- DCB header does not point to DCB.
- DSORG in DCB is not VIP or VSP.
- Type-R STOW, member name not found, and not called by ABEND.
- DCB already opened for another member.
- A call to Search SDST gave an error return.
- Member checked out by another DCB.
- DCB not linked to specific member.
- Member in use; cannot delete it.

When any of the following errors are detected, a VDMEM is executed and the function (not the task) is terminated:

- Read-only access.
- No external storage space.
- Storage ration exceeded.
- No secondary allocation.
- Attempt to expand RESTBL for shared data set.
- Maximum data set size exceeded.
- POD/RESTBL mismatch.
- Maximum directory size exceeded.
- Invalid return code from CZCOM or CZCOG (will also cause SYSER).
- Duplicate input STOW NA or NAR. Invalid or closed DCB passed as parameter.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the Stow CSECT and PSECT, ISA, DCB, DCB header, member header, and POD.

The validity of the STOW type, and the validity of particular types of STOW with the existing OPEN options, are verified. ABEND is called if any errors or inconsistencies are found.

The RESTBL is interlocked for shared data sets. The lock will be released prior to any RETURN VSAM CLOSE (CZCOQ) or VISAM CLOSE (CZCQA) are called if required.

The POD is searched to locate the member descriptor. If the member name is not found in the POD, a member descriptor is built for a type-N STOW. The POD and RESTBL page counters are updated to reflect the addition of the new member and to reflect changes to all OPENed members. If the POD has been updated by STOW, the POD Integrity flag in the RESTBL is set; this causes CLOSEVAM (CZCOB) to write the POD to external storage. Control is returned to the caller by the RETURN macro.

If the member name is found in the POD, processing continues for the particular STOW type:

- Type-N -- If a name was found on a type N (new member), no further processing is done, and return is made with a return code signifying that the new name was not unique.
- Type-NA -- New aliases may be added to an existing member. The POD is searched for each alias being added. If the aliases are unique, an alias descriptor is created for each alias being added. These new alias descriptors are linked to the appropriate member descriptor, and control is returned to the caller by the RETURN macro.
- Type-R -- STOW type-R is called to replace user data, and close the member. If user area was specified, the user data is stored in the POD. This may cause the member descriptor to be moved, since it may not previously have contained user data, or the new user data requires additional space. If the member descriptor is moved, the hash pointers and the alias links are updated. If the user count in the member header is zero, the member header is closed, (that is, the member header is added to the deleted member header chain, the DCB header link is cleared, and the member name is set to 8XL1'FF'). The POD is updated, and control is returned to the caller by the RETURN macro.
- Type-U -- Type-U STOW accomplishes the same thing as type-R, except that the member header is not closed. It remains active for further processing.
- Type-D -- STOW type-D (delete) causes the data pages associated with a member to be deleted by calling Reclaim (CZCOG). The member and alias descriptors are deleted from the POD. The DCB is initialized for reuse, and control is returned to the caller by the RETURN macro.
- Type-DA -- Aliases may be deleted from an existing member. The POD is searched for the alias being deleted. If found, the alias descriptor is deleted from the POD. This process is done for each alias being deleted.
- Type-C and Type-CA -- Type-C and -CA STOW are name changes of members or aliases. The POD is searched for the name being changed. When found, the new name member or alias replaces the old name accordingly.
- Type-NAR -- New aliases may be added to an existing member as in a type-NA STOW but if any new aliases duplicate existing aliases or names, none are stowed and a list of these duplicates is supplied to the caller. Stow does a GETMAIN for VMA of list, passes VMA in register 0 and will not FREEMAIN this area. If there are duplicate aliases within an input list of new aliases,

some aliases may be stowed and, although a return code of '04' will be set, no duplicate names will be supplied the caller (see Restrictions).

Search Routine (CZCOL)

Search is called by Find, Stow, and GETNUMBR to locate a member descriptor in the POD for a given member name or alias. MOSEARCH, entered at Search's second entry point, may be used to search past the first matching entry in the POD for additional entries with the same name. (See Chart PC.)

Attributes: Read-only, reenterable, privileged, public, system.

Restriction: If MOSEARCH is called, Search must have been called first, and the results of that search left undisturbed.

Entry Points:
CZCOL1 -- Entered via type-1 linkage to locate first matching member descriptor.

CZCOL2 -- Entered via type-1 linkage to search past first matching entry in the POD, seeking additional entries with the same name.

Input: General register 1 contains the address of a two-word parameter list:

Word 1 -- Address of the DCB associated with the POD to be searched.

Word 2 -- Address of an 8-character field containing the member name or alias to be used as a search key.

The search code field in the DCB will contain a code to indicate the type of search being requested:

M	Member
A	Alias
E	Either

Modules Called: None.

Exits:
Normal -- Register 15 contains one of the following return codes:

'00' Successful.

'04' Entry with matching name and type not found -- no hashing chain for hash value.

'08' Entry with matching name and type not found -- hashing chain exists for hash value.

Error -- None.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the CSECT, DCB, DCB header, POD, and POD member descriptor.

The DCB is modified per results of Search or MOSEARCH:

DCB

Symbol	Description
DCBHV	Hash value of member or alias.
DCBSC	Relative location within POD of found descriptor.
DCBSP	Relative location within POD descriptor of preceding found descriptor in same hash chain.

Search has two entry points. CZCOL1 is the entry point for the initial call to Search. CZCOL2 is the entry point for continuing Search (MOSEARCH), after Search has initially been called. A switch (DCBSWT) is set to indicate that the current entry was to Search (CZCOL1). This switch is tested on every entry; if on, the member name is hashed; if off, the name is not hashed, since it was hashed on the initial entry and the value still exists in the DCB (DCBHV).

The hash value is used to obtain the hash chain pointers from the POD. If the pointer is zero, the return code is set to "no hash chain", and control is returned to the user by the RETURN macro.

If the hash chain pointer exists, the descriptor is obtained. The name input to Search is compared to the descriptor name. If the names do not compare, the rest of the chain is searched in the same manner until the name is found or the end of the chain is reached. If the end of the chain is reached and the name is not found, the return code is set "not found" and control is returned to the caller by the RETURN macro.

If a descriptor name is found that corresponds to the Search input name, the search type (member, alias, or either) is tested to determine if the found descriptor is the correct type. If the search type was "E" (either), the search was successful, and the return code is set to "found." The location of the last two descriptors are saved (for MOSEARCH) and control is returned to the caller by the RETURN macro.

If the search type was "M" (member), and the found descriptor is a member descriptor, a "found" exit is made as described above. If the search type was "M" and the

found descriptor is an alias, the search of the hash chain is continued.

If the search type was "A" (alias) and the found descriptor is an alias, a "found" exit is taken as described above; otherwise, the hash chain search is continued.

Extend POD Routine (CZCOM)

Extend POD is called by Stow to expand the POD by one page, both in virtual storage and on the external storage device. (See Chart PD.)

Attributes: Read-only, reenterable, privileged, public, system.

Restrictions: It is not possible to expand shared data tables. If this becomes necessary, a return code is passed to the caller and this function terminates.

Entry Point: CZCOM1 -- Type-1 linkage (privileged to privileged).

Input: Register 1 contains the address of the DCB associated with this POD.

Modules Called:
Expand (CZCG4) -- Expand the size of the virtual storage area containing the POD.

Insert (CZCOF1) -- Insert external page entry into RESTBL.

Relocate Members (CZCON1) -- Adjust member page numbers to compensate for expanded POD.

VDMEP (CZQK1) -- Output a diagnostic message and terminate the function (but not the task).

Exits:
Normal -- Return to the calling routine with one of the following return codes:

'00' Normal.

'04' No storage space available.

'08' Storage ration exceeded.

'0C' No secondary storage allocation specified.

'10' Shared data set RESTBL cannot be expanded.

'14' Maximum data set/or member size exceeded.

'18' Insertion beyond end of data set.

'1C' Deletion beyond end of data set.

Error -- VDMEP is called if an invalid return code is received from Insert.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the CSECT and PSECT, DCB, DCB header, POD and RESTBL.

The POD is expanded by one page by calling Expand with the virtual storage address of the POD, the number of pages currently in the POD, and a one page expansion request.

On return from Expand, the returned virtual storage address is tested to see if the POD was relocated due to expansion. If relocated, the POD base and POD pointer in the RESTBL have to be updated. An external page entry is inserted in the RESTBL (by calling Insert) to correspond to the new POD page. Error returns from Insert cause an ABEND. The data set length is checked to see that the new page added does not exceed the maximum allowable size of a partitioned data set. A return code is set if the maximum data set length has been exceeded.

The number of POD pages is updated in the POD and RESTBL.

Since the page inserted in the data has changed the relative position of all members, Relocate Members (CZCON) is called to update the relative location of existing members in the POD.

Control is returned to the caller by the RETURN macro.

Relocate Members Routine (CZCON)

Relocate Members is called by GETNUMBR and STOW, to update member descriptors in the POD to compensate for added or deleted pages within a partitioned data set. Member headers of members that are checked out are also updated. (See Chart PE.)

Attributes: Reenterable, read-only, privileged, public, system.

Entry Point: CZCON1 -- Entered via type-1 linkage.

Input: Register 1 contains the address of the DCB associated with the data set being modified. Relevant fields in the DCB are set as follows:

DCBN Data set page number at which relocation must occur.

DCBM Number of pages inserted or deleted.

DCBOP '0800' indicates insert.

'0400' indicates delete.

Modules Called: None.

Exits: Return to the calling routine.

Operation: Initialization and general register storage is executed in conformance with linkage conventions. Base registers are declared for the CSECT, DCB, DCB header, POD, Member Header, and RESTBL.

Member descriptors in the POD are located by searching the hash chains. They are examined to determine if they have been relocated. The relative location and the external page value of the first page of each member is recorded in the POD.

If a member requires relocation (that is, it is past the point of relocation), the member is tested to determine if it is the member causing relocation and if the first page is affected. If the first page is affected, the first page value is updated and the search of the hash chain is continued.

If the current member is not the member causing relocation, or it is the member causing relocation and the first page is not affected, the POD is updated to reflect location.

The member header chain in the RESTBL is searched to determine if the current member is checked out (that is, active - a member header exists). If the member is checked out, the member header is updated to reflect the relocation. The search of the hash chain is resumed.

When a hash chain is exhausted, the next hash chain is searched until all chains have been processed. The member header chain is tested to see if any members are being created. The start of members being created is adjusted in the member header for any relocation.

Control is returned to the caller by the RETURN macro.

GETNUMBR Routine (CZCOO)

GETNUMBR (Get Member Page Number) is called by MOVEPAGE (CZCOC) to convert the page number relative to member, to the page number relative to the data set; and by Insert/Delete Page (CZCOD) to control changes in the size of a member. (See Chart PF.)

Attributes: Read-only, reenterable, privileged, public, system.

Entry Point: CZCOO1 -- Via type-1 (privileged to privileged) linkage.

Input: Register 1 contains the address of the DCB associated with the partitioned data set for which member page numbers must be corrected. Relevant fields in the DCB are:

DCBN Page number relative to member of the first page in the request.

DCBM Number of pages to be processed.

DCBOP '8000' - Input.

'2000' - Output.

'0800' - Insert.

'0400' - Delete.

Modules Called:

Search (CZCOL1) -- Search POD for member descriptor.

Insert (CZCOF1) -- Insert pages in RESTBL.

Relocate Members (CZCON1) -- Adjust member description for added or deleted pages.

Reclaim (CZCOG1) -- Delete member page entries from RESTBL.

TSEND (CEAH19) -- Wait for shared pages to go out of use before deleting pages.

VDMEP (CZCQK1) -- Output a diagnostic message and terminate the function (but not the task).

Exits:

Normal -- Register 15 contains one of the following return codes:

'00' Successful.

'04' No external storage space available.

'08' Storage ration exceeded.

'0C' No secondary storage allocation specified.

'10' Shared data set RESTBL cannot be expanded.

'14' Maximum data set or member size exceeded.

'18' Insertion beyond end of data set.

'1C' Deletion beyond end of data set.

Error -- VDMEP is called if an invalid return code is received from Reclaim or

insert, or if an old member could not be located in the POD (the member has been deleted abnormally).

Operation: Initialization and general registers are stored in conformance with linkage conventions. Base registers are declared for the CSECT and PSECT, DCB, DCB header, member header, POD and RESTBL.

The offset of the member is tested to determine if it has been relocated since the last operation. An adjustment has to be made in the POD and member header for an old member. Only the member header need be adjusted for a new member, since it has not been STOWed and no POD entry exists.

The extent of the current operation is tested to see if it is within the range of the member for a deletion or pure number translation, or contiguous to the member for an insertion. A return code is set if the operation is not within the computed limits.

The page number relative to the member is converted to a page number relative to the data set. If the operation is not an insertion or deletion, control is returned to the caller by the RETURN macro.

If the operation is a deletion, the pages are deleted by calling Reclaim (CZCOG). If the deleted pages did not belong to a member being created but to an existing member, Relocate Members (CZCON) is called to relocate the member in the POD and member headers, by the amount of deletion. Control is then returned to the caller by the RETURN macro.

Similarly, the data pages are inserted by calling Insert (CZCOF). Relocate Members is called if the insertion was in an existing member, to adjust the other members by the amount of the insertion. Control is returned to the caller by the RETURN macro.

PART III

QUEUED SEQUENTIAL ACCESS METHOD (QSAM)

QSAM routines operate upon data sets of queued, sequential organization. They will process all of the OS/360 QSAM facilities. (QSAM uses move-mode to provide the functional equivalent for OS/360 substitute-mode programs.) In addition, TSS QSAM provides the following features not supported by OS/360 QSAM:

1. Both locate and move mode macro instructions can be intermixed on the same data set.

2. Variable record formats are allowed on a data set opened for RDBACK.

3. A SETL routine is provided to alter sequential processing of a QSAM data set.

QSAM's basic functions are blocking and deblocking logical records, issuing I/O requests, checking, and positioning for blocks of data.

QSAM itself blocks, deblocks, and buffers internally, but uses BSAM to perform I/O operations such as reading, writing, checking, and positioning for access to data. Through BSAM routines, QSAM also provides labeling services and, if required, ASCII translation. Table 49 lists the modules of BSAM invoked by QSAM, and briefly describes their functions.

Table 49. Usage of BSAM Modules

Title and Module ID	VCON	RCON	Usage
READ/WRITE CZCRA	CZCRAS	CZCRAP	Reads or writes blocks of data
CHECK CZCRC	CZCRCS	CZCRCP	Checks the completion of read or write operations
POINT CZCRM	CZCRMA	CZCRMP	Repositions a data set
CNTRL CZCRB	CZCRBS	CZCRBP	Repositions a data set
NOTE CZCRN	CZCRNA	CZCRNP	Returns relative address within a volume of last block read or written
BSP CZCRG	CZCRGA	CZCRGP	Backspaces a data set

QSAM Macro Instructions

The macro instructions used on a QSAM data set fall into three groups:

1. Those which are directly serviced by QSAM.

   a. The GET macro instruction retrieves for the user a single logical record.

   b. The PUT macro instruction adds a single logical record to a block of records.

   c. The PUTX macro instruction returns an updated block of records to a data set, or includes a record of an input data set in an output data set.

   d. The TRUNC macro instruction causes the next logical record of an output or update data set to be treated as the first record of the next block.

   e. The RELSE macro instruction causes the remaining logical records in an input or update buffer to be ignored.

   f. The SETL macro instruction specifies a new start location for sequential processing.

2. Those which are serviced only by BSAM, but affect the operation of QSAM.

   a. The OPEN macro instruction fills in certain fields of the data control block that were not filled in at assembly time, checks volume labels, constructs tables, and provides work space and buffer areas.

3. Those which are serviced mainly by BSAM with additional functions performed by QSAM.

   a. The CLOSE macro instruction completes or purges all outstanding I/O requests, releases the storage obtained by the OPEN routines, and writes trailer labels.

6. The FEOV macro instruction directs the control program to advance to the next volume of a data set.

Work Area and Buffers

QSAM expects SAM Open and SAM Close to perform several functions concerning acquisition of working space. Because it is assigned no PSECT, QSAM requires a work area. SAM Open obtains the space for this work area and also provides buffer areas for QSAM.

The work area provided for QSAM is known as the QWK work area, or QWKAR. Its address is placed in the DCBQWK field of the data control block, by SAM Open. All fields within the QWK work area, except those words reserved for data event control blocks, are referred to with a prefix of QWK. QWKAR consists of:

1. A 19-word save area (QWKLEN-QWSI3).
2. Sufficient storage for three data event control blocks (DECB1-DECB3).
3. An 8-word save area for saving return addresses between subsections of QSAM (QWKGR0-QWKGR7).
4. Another 8-word save area for saving register contents when EODAD or SYNAD is invoked (QWKWK1-QWKWK8).
5. Another 11-word save area for saving registers in a type-2 linkage.

The DSECT used to refer to QWKAR can be found in Appendix A.

In addition to providing the QWK work area, SAM Open will also provide QSAM with:

1. One buffer each for data sets opened for Update, and when SETL (S) is requested in the DCB MACRF field.
2. Three buffers for each data set opened for RDBACK using variable record format.
3. Two buffers each for all other data sets.

QSAM's buffering techniques will be discussed further in Part II of this section.

Control Blocks

Following are brief descriptions of each of the control blocks used by QSAM.

Data Control Block (DCB): The DCB is QSAM's primary source of information about the data set. It is defined by the user at assembly time through the DCB macro instruction, and may be filled in or modified at OPEN time or during execution. Appendix B contains a list of the main fields of the DCB used exclusively by QSAM with a brief description of each.

Data Event Control Block (DECB): The DECB provides information necessary for the control of each I/O operation and reflects the status of the completed operation. QSAM initializes the DECB before I/O requests are passed to BSAM, and the BSAM Posting routine completes the control block.

Data Extent Block (DEB): QSAM uses the DEB to determine if there are any outstanding, unchecked I/O requests or error conditions.

It is important to note that QSAM is designed as a "fence sitter" routine, and will run in the same privileged status as the routine which invokes it. Also, since no linkage is established to the problem program when QSAM has been invoked by any other routine, all linkage between the problem program and QSAM will be of type-1.

There are 22 subroutines within the QSAM module. Since QSAM does not have a PSECT, register usage and register saving are kept to a minimum by carefully regulating use of those registers.

The linkage between subroutines differs from normal linkage procedures, because when one subroutine is invoked by another, it may not return directly to the invoking subroutine without having first invoked one or more other subroutines. The normal procedures for invoking a subroutine, with the return address in register 14, require a separate save area for each of the 22 subroutines, in order to maintain the integrity of return addresses. The 22 subroutines are therefore divided into seven levels, such that no subroutine invokes another subroutine on the same level, either directly or indirectly. For example, the GET subroutine on level 3 never invokes another subroutine which is also on level 3, nor invokes any subroutine which in turn invokes another subroutine on level 3. Thus one register is assigned as a return register for all subroutines on one level, there being no need for a separate return register save area for each level. Thus, each subroutine has an exit register based upon its level, as defined in the subroutine interface table (Table 50).

All subroutines are invoked by a BASR instruction with register 15 containing the entry point address of the subroutine, and the exit register of the particular subroutine containing the return address. The invoked subroutine must save its exit register in one of the fields (QWKGR1-QWKGR7) provided in the QWK work area for this purpose, in order that the exit register may also be used for calculation during processing. When it has completed its processing, the subroutine restores its exit register and issues a Branch to Register instruction (BR).

QSAM uses two base registers, which are established at each of the nine entry points. Another register, the base register for the DCB known as DCBREG, is also established at each entry point, and remains the same throughout the processing of any one macro instruction. There is also an assigned base register for the DECB, known as the DECREG, which must be loaded by each subroutine using it, since there may be more than one DECB in use. Registers 0 and 1 are also reserved to pass parameters between subroutines. The parameters expected by each subroutine are listed in Table 50.

Since QSAM generally runs in the same privileged state as the problem program, it may or may not be of the same privilege as the BSAM modules which it invokes. All of the modules listed in Table 49, except CZCRN, are privileged routines. CZCRN is also constructed as a "fence-sitter" routine, and will take on the privilege status of QSAM whenever it is invoked by QSAM. Therefore, type-1 linkage is always established to invoke CZCRN, using the V-con and R-con defined within the QSAM module.

Before establishing linkage to any of the other BSAM modules, it is necessary to determine the status of QSAM. The subroutines Read/Write, Check, Point, Control, and Backspace perform this function with respect to their BSAM counterparts, by testing the first bit of the VPSW in the ISA table (CHAISA). If QSAM is privileged, type-1 linkage is established using the V-cons and R-cons defined within the QSAM module. If it is not privileged, type-2 linkage is established via the ENTER SVC, with the appropriate code in register 15.

The parameters expected by the BSAM modules and the possible return codes from them are listed in Table 51.

QSAM Routine (CZCSA)

QSAM blocks and deblocks logical records within a buffer, performs buffering services, and issues requests to BSAM for transfer of data between storage and any I/O device. (See Chart QA.)

Attributes: Reentrant, nonrecursive, closed, resident in virtual storage, assumes privilege of caller.

Entry Points: QSAM has nine entry points and nine entry sections which serve to channel processing through the appropriate subroutines.

CZCSAA -- Entered upon issuance of the first GET on a data set, the first GET following a SETL type-E or -B, or the first GET following a FEOV.

Table 50. Subroutine Interface

Subroutine	Exit Register	Parameter Registers on Entry	Other Subroutines Invoked
GET	3	1=DCB address 0=user work area address (if any)	GETIO, PUTXIO
PUT	3	1=DCB address 0=user work area address (if any)	PUTIO
PUTX	2	1=DCB address 0=Input DCB address for output PUTX	PUT, PUTXIO
TRUNC	2	1=DCB address	PUTIO
RELSE	2	1=DCB address	-
TREOV	3	1=DCB address	PUTIO, CHECK, FLUSH
SETLR	2	1=DCB address 0=pointer to TTRZ or ZZCC*	TREOV, INITIO, GET, CHECK, POINT, BSP, CNTRL
SETLP	2	1=DCB address	BSP, INITIO, GET, SYNAD, CNTRL, TREOV, FLUSH
GETIX	2	1=DCB address	CHECK
SETFR	2	1=DCB address	TREOV, CNTRL, POINT
INITIO	3	1=DCB address	GETIO, READ/WRITE
GETIO	5	1=DCB address	READ/WRITE, CHECK, COMIO
PUTIO	4	1=DCB address	READ/WRITE, CHECK, COMIO
PUTXIO	4	1=DCB address	READ/WRITE, CHECK, GETIO
COMIO	6	1=DCB address	-
SYNAD	7	1=error type code 0=DECB address	-
READ/WRITE	6	1=DECB address	-
CNTRL	3	1=DCB address 0=action code and value	SYNAD
BSP	3	1=DCB address	-
POINT	3	1=DCB address 0=pointer to TTRZ or ZZCC*	SYNAD
CHECK	6	1=DECB address	SYNAD
FLUSH	4	1=DCB address	CHECK

*The terms TTRZ and ZZCC refer to the relative form of the retrieval address of any block within a data set on magnetic tape or direct access devices. This address is obtained, in its relative form, by BSAM NOTE (CZCRN). TTRZ refers to data sets on magnetic tape, and ZZCC refers to data sets on direct access devices.

Table 51. Parameters and Return Codes of BSAM Modules

Table 51. Parameters and Return Codes of BSAM Modules

Module	Parameters	Return Codes
CZCRA	GR1=DECB address	None
CZCRC	GR1=DECB address	SYNAD request flag in DECB EODAD flag in DCB GR0=DECB address
CZCRM	GR1=DCB address GR0=pointer to TTRZ or ZZCC	Normal return, GR15=0 Error return, GR15=4 Unrecoverable error flag in DEB
CZCRB	GR1=DCB address GR0=action code and number value	Normal return, GR15=0 Error return, GR15≠0
CZCRG	GR1=DCB address	Normal return, GR15=0 Error return, GR15≠0
CZCRN	GR1=DCB address	GR1=TTRZ or ZZCC address

CZCSAB -- Entered upon issuance of the first PUT on a data set, the first PUT following a SETL type-E or -B, or the first PUT following a FEOV.

CZCSAG -- Entered upon issuance of all GETS except those listed under CZCSAA.

CZCSAW -- Entered upon issuance of all PUTs except those listed under CZCSAB.

CZCSAX -- Entered upon issuance of a PUTX.

CZCSAT -- Entered upon issuance of a TRUNC.

CZCSAR -- Entered upon issuance of a RELSE.

CZCSAV -- Entered only by SAM Close or FEOV.

CZCSAS -- Entered upon issuance of a SETL.

Input: The following parameters are passed:

Register 0 -- Address of work area (if any) for entries CZCSAA, CZCSAB, CZCSAG, and CZCSAW.

Register 0 -- Address of input DCB for CZCSAX when an output-mode PUTX is issued.

Register 1 -- Address of DCB for all entry points.

Data References: CHADCB, CHADEC, CHADEB, CHAISA, QWKAR.

Modules Called:
BSAM Read/Write (CZCRA) -- Entry at CZCRAS. For data transfer.

BSAM Check (CZCRC) -- Entry at CZCRCS. Test I/O results.

BSAM Point (CZCRM) -- Entry at CZCRMA. Reposition a data set.

BSAM Control (CZCRB) -- Entry at CZCRBS. Reposition a data set.

BSAM Backspace (CZCRG) -- Entry at CZCRGA. Backspace.

BSAM Note (CZCRN) -- Entry at CZCRNA. Identify last record read or written.

Exits:
Normal -- Return to the calling routine.

Error -- ABEND termination under the following conditions:

a. During processing of a GET macro instruction, when the computed sum of the logical record lengths (11) of variable length records does not equal the specified block size (LL).

b. During processing of a PUT macro instruction, when the user attempts to PUT a logical record longer than the specified maximum block size.

c. During processing of a PUT macro instruction, when the user specifies a value in the length control bytes (11) of a variable record larger than the logical record length previously estimated in the DCB.

d. During processing of a PUTX macro instruction, when the previous macro instruction was not a locate-mode GET.

e. During processing of an output-mode PUTX macro instruction, issued on an output DCB whose address is in register 1, when the associated DCB, whose address is in register 0, has been opened for Output.

f. During construction of a block of fixed format records, if the user causes an incorrect length output block to be created by changing the value of the logical record length.

g. When a PUT macro instruction is issued on an update, input, or read-back data set, or when a GET macro instruction is issued on an output data set.

h. When a SETL is issued, but there is no (S) in the DCB MACRF.

Otherwise, exit to user's SYNAD or EODAD routine.

Operation: The subroutine functions are shown in Table 52.

Table 52. Subroutine Functions

Name	Entry(s)	Chart	Function
GET	CZCAS7	DM	Deblocks logical records
PUT	CZCSA8	DN	Blocks logical records
PUTX	CZCSA2	DO	Returns logical records retrieved by a locate-mode GET to an UPDATE or OUTPUT data set
TRUNC	CZCSA3	DP	Truncates current block (output or update)
RELSE	CZCSA4	DQ	Releases current block (input, update, or readback)
TREOV	CZCSA9	DR	Completes or purges outstanding I/O requests
SETLR	CZCSAL	DS	Positions data set at specified retrieval address
SETLP		DT	Positions data set at previous logical record
SETLC	CZCSAZ	DU	Obtains retrieval address (TTRZ or ZZCC) of current logical record
SETLEB	CZCSAH	DV	Positions data set at beginning or end of current volume
INITIO	CZCSA6	DW	Initializes buffer addresses, block size, etc., and constructs DECBs
GETIO	CZCSA5 (Entry from GET)		Performs buffering for input operations
	CZCSAI (Entry from PUTX)		
PUTIO	CZCSAU		Performs buffering for output operations
PUTXIO	CZCSAJ		Performs buffering for update data sets
COMIO	CZCSAM	DX	Initializes for a new buffer
SYNAD	CZCSAN		Transfers control to user's SYNAD routine and performs checks on error options
READ/WRITE	CZCSAD (Entry from GETIO)		Issues an I/O request for data transfer
	CZCSAE (Entry from PUTIO)		
CNTRL	CZCSAC		Requests repositioning of data set
BSP	CZCSAY		Requests backspacing of one block
POINT	CZCSAP		Requests repositioning of data set
CHECK	CZCSAK		Requests a check on results of an I/O operation
FLUSH	CZCSAF		Purges I/O activity from DECB queue

Blocking Logical Records: The user issues a PUT macro instruction for each logical record he wishes to include in the output data set. The PUT subroutine adds the logical record to the block if it will fit within the current buffer. Otherwise, the block is considered complete, and the record for which the PUT was issued will be treated as the first record of a new block. The user can cause a block to be regarded as complete prematurely by issuing a TRUNC macro instruction.

Deblocking Logical Records: The GET subroutine returns to the user a single logical record each time he issues a GET macro instruction. When a block of records has been read and checked, the buffer address of the first logical record is returned to the user if the GET macro instruction was in locate mode; or, if it was in move mode, the first logical record is moved to his work area. When the current block is completely processed, the next GET issued causes the buffer to either be refilled if the data set was opened for Input or Rdback, or to be written back, if required, to an update data set and then refilled. At any time, the user can cause processing on a buffer to be regarded as complete by issuing a RELSE macro instruction.

Buffering Blocks of Data: The normal buffering facility of QSAM is known as double buffering. This involves the use of two buffers, one of which will be currently in use while I/O activity is being performed on the other. Thus, on a normal input or readback data set, while logical records from one buffer are being supplied to the user, the other buffer is being refilled. On a normal output data set, QSAM will continue adding logical records to one buffer while the other is being written out.

Each buffer is assigned to one of the first two DECBs contained in QSAM's QWK work area. Pointers to these DECBs are contained in the DCB (DCBDE1 and DCBDE2). To achieve the alternating of buffers, all read or write operations are performed on the DECB pointed to by DCBDE2, and all checking operations are performed on the DECB pointed to by DCBDE1. Following each checking operation, the two pointers are switched so that, when the next I/O is initiated, the read or write performed on the DECB previously pointed to by DCBDE2 and now pointed to by DCBDE1 will be checked, and the buffer belonging to the DECB now pointed to by DCBDE2 will be either refilled or written out.

Once a buffer has been either filled or written out, and checked, it is available for processing, and QSAM will begin returning logical records from it to the user, or adding logical records to it as they are supplied by the user. When this current buffer is completely processed, QSAM issues either a read request to refill it, or a write request to write it out. Then the previous read or write operation is checked, and that buffer becomes available for processing. Buffering is performed for input or readback, output, and update data sets by the GETIO, PUTIO, and PUTXIO subroutines, respectively.

Under some circumstances, it is necessary to perform only single buffering; that is, only one buffer is used. In this case, the pointers to the two DECBs are both set to point to the first DECB, so all operations will be performed on the same DECB regardless of the switching of pointers.

The decision to use double or single buffering is based on the OPEN option of the data set, or on the combination of device type and macro option specified in the DCB. Double buffering will be done in all cases except the following:

1. When the data set is opened for UPDATE.

2. When the DCB MACRF requests a SETL (S).

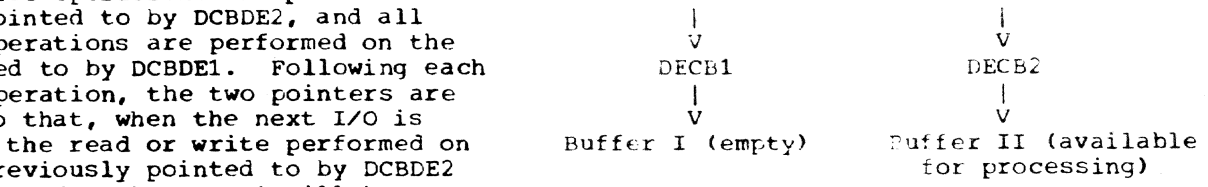
Single buffering must be done on an update data set to allow the user to update one block of records at a time. No reading ahead can be done until it is determined whether or not the current block of records must be updated, since an update write can only return the last block read.

With double-buffering facilities, QSAM requests two writes before requesting a check on the first write.

Examples of Double Buffering:

I. Double buffering involving an output data set.

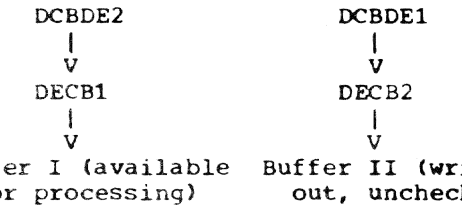
Phase I



QSAM builds the first block of the user's data set in Buffer II by adding to the buffer each logical record for which a

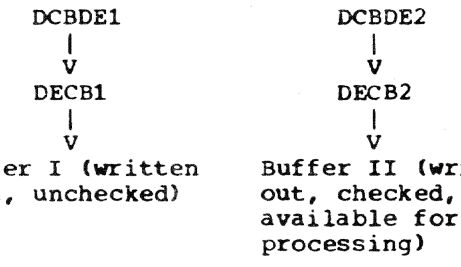
PUT is issued, until the buffer is full. Then a write request is issued for Buffer II, a check request is issued on Buffer I (since DECB1 is initialized by QSAM to indicate normal completion, this check constitutes a dummy request), and the DECB pointers are switched.

Phase II



QSAM now builds the second block of records in Buffer I. When it is complete, a write request is issued for Buffer I, a check request is issued for Buffer II, and the DECB pointers are switched.

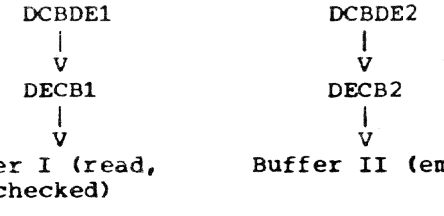
Phase III



Buffer II is again available for processing. The third block of the data set is built in Buffer II, and when complete, a write request is issued for Buffer II, a check request is issued for Buffer I, and the DECB pointers are again switched. Thus the processing operation continues, alternating the buffers used, until the user has placed the last logical record of his data set in the buffer, at which time he may CLOSE the data set, causing the last block of records to be written out and checked immediately.

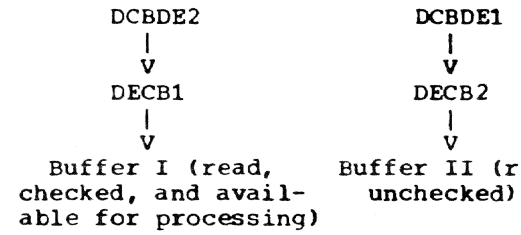
II. Double buffering involving an input data set.

Phase I



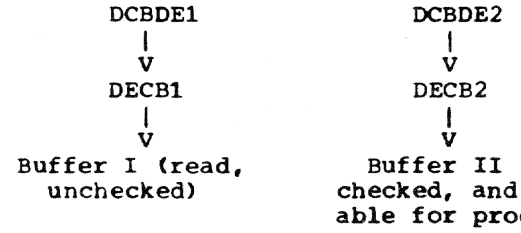
Since on the first I/O request of an input data set, both buffers must be primed, an initial read is requested on Buffer I. (This varies from the normal procedure of requesting reads only on the DECB pointed to by DCBDE2.) Then another read is requested to fill Buffer II, a check is requested on Buffer I, and the DECB pointers are switched.

Phase II



Each of the logical records in Buffer I is returned to the user when he issues a GET macro instruction. When all the records in Buffer I have been returned to him, a read request is issued to refill Buffer I, a check request is issued on Buffer II, and the DECB pointers are switched.

Phase III

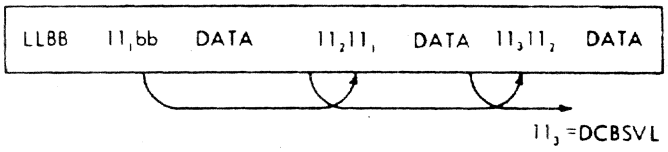


Buffer II is now the current buffer in use, and each of the logical records in it is returned to the user until there are none left, at which point Buffer II will be refilled, Buffer I will be checked, and the DECB pointers will be switched again. Processing continues in this manner until an end of data set is encountered. At that point, the user's EODAD routine gains control.

III. Double buffering on a readback data set.

Double buffering on a readback data set of fixed or undefined length records is handled in the same manner as that on an input data set, except that blocks of records are read beginning with the last block of the data set. However, if a data set opened for Rdback specifies variable-format records, the procedure is varied to include the use of a third buffer. After a block of records has been read and checked, a copy of it is moved to the third buffer. This copy is used as a table to contain record lengths so that the records contained in the actual buffer may be accessed

in reverse order. A forward search is made through the block for the length control bytes (11) in front of each record, and these bytes are saved in the blank control bytes (bb) of the following record. The length of the last record in the block is saved in the DCB in order that the address of the first record to be accessed (logically the last record of the block) may be computed. The following diagram demonstrates the layout of the third buffer.



The beginning-of-buffer address for blocks of records in a readback data set is at the logical end of the block. By subtracting from that address the length of the last record in the block, which is saved in the DCB, the first record requested by the user is accessed from the actual buffer. Then the length of the preceding record is obtained from the control bytes of the current record in the third buffer, and saved in the DCB so that the address of the next record may be computed. By maintaining its own copy of the current buffer, QSAM can be certain the lengths of the previous logical records recorded in the control bytes are always correct.

Issuing I/O Requests, Checking, and Positioning for Blocks of Data: QSAMs internal functions are performed entirely within storage. Any I/O requests for transfer of data between storage and any I/O device, or requests for repositioning a data set, are passed on to BSAM. Table 50 in this section lists the BSAM modules invoked by QSAM and the expected return codes.

Before discussing the internal logic of the QSAM routine in relation to QSAM macro instructions, it is necessary to establish certain rules which will be followed throughout the discussion.

1. All fields within the DCB, when referenced, have a prefix of DCB.

2. All fields within the DECB, when referenced, have a prefix of DEC.

3. All references to fields within either table indicate the contents of the field unless it is specifically stated otherwise.

4. Each of the nine entry subroutines is referred to by the name of its entry point. A calling module always enters QSAM at the entry point itself. For internal processing, however, a reference to "returning to CZCSAA" will indicate the entry subroutine, not the entry point.

5. Since certain of the QSAM subroutines have the same names as BSAM modules, such as Read/Write or Check, reference to the BSAM module of that name will always be clearly marked by referring to "BSAM Read/Write (CZCRA)" or by using just the module ID "CZCRA" after it has been identified with the name of the module.

COMMON PROCESSING

Certain of the QSAM subroutines perform the same function each time they are entered, regardless of the type of data set, device or macro instruction being used. To simplify the tracing of each macro instruction, these subroutines are briefly described below, and thereafter referred to only to indicate the specific points at which they are invoked to perform their functions.

SYNAD Subroutine

SYNAD is entered whenever an error in reading, writing, or positioning has occurred. If the user has provided his own SYNAD routine, a copy of the DECB on which the error occurred is moved to DECB3, and control is passed to his SYNAD. If not, or upon return from the user's SYNAD, tests are made to see if the erroneous block may be accepted or skipped, or if the task must be abnormally terminated. If no abnormal exit is required, the appropriate flags are set in the DCB, and a return is made to the calling subroutine.

Read/Write Subroutine

The Read/Write subroutine has two entries. CZCSAD is the entry when a read request is issued, and CZCSAE is the entry when a write request is issued. The appropriate type code is set into the DECB, and linkage is established to BSAM Read/Write. When control is returned to Read/Write, a return is made to the calling subroutine.

Control Subroutine

The Control (CNTRL) subroutine invokes the BSAM Control routine. If the return code from BSAM Control is not zero, the SYNAD subroutine is invoked. Otherwise, a return is made to the calling subroutine.

Backspace Subroutine

The Backspace subroutine invokes BSAM Backspace. After control is returned to Backspace, a return is made to the calling subroutine.

Point Subroutine

The Point subroutine invokes BSAM Point. On return from CZCRM, the SYNAD subroutine is invoked if the return code register contains a four. If the return code is zero, or upon return from SYNAD, a return is made to the calling subroutine.

Check Subroutine

The Check subroutine establishes linkage to BSAM Check (CZCRC). After control is passed back from CZCRC, if SYNAD is requested the SYNAD subroutine is invoked, and after control is passed back to Check, a return is made to the calling subroutine. If SYNAD is not requested and EODAD is indicated, control is passed to the user's EODAD routine, from which no return is expected. Otherwise, a return is made to the calling subroutine.

Flush Subroutine

Three cases exist:

1. If no I/O requests are outstanding, an exit is made to the calling subroutine.

2. When any I/O request is outstanding and the DECB has been marked intercepted and EODAD has been requested, then the number of outstanding I/O requests in the DEB (DEBNCP) is set to zero, and a return is made to the calling subroutine.

3. If the I/O request which remains outstanding is not complete, an AWAIT SVC is issued and the completion is awaited. The Purge flag in the DECB is then set on, and control is passed to the Check subroutine. Upon return from Check, a return is made to the calling subroutine.

GETIO Subroutine

The GETIO subroutine first invokes Read/Write to perform a read on the DECB pointed to by DCBDE2, and then invokes Check to check the read done previously on the DECB pointed to by DCBDE1. If Check detected an error in the read operation and the user indicates it is to be skipped, GETIO switches the DECB pointers and goes back to repeat the reading and checking operations, until no error is detected. Otherwise, GETIO invokes the COMIO subroutine to initialize buffer addresses, and then returns to the calling subroutine.

PUTIO Subroutine

PUTIO invokes Read/Write to perform a write on the DECB pointed to by DCBDE2, and then invokes Check to check the DECB pointed to by DCBDE1. PUTIO gives control to COMIO to reinitialize the free buffer, and then makes a return to the calling subroutine.

PUTXIO Subroutine

IF A PUTX has been issued on the current block, PUTXIO invokes Read/Write to write the updated block back to the data set, and Check to check the completion of the write. Following this, or if no PUTX was issued on the block, PUTXIO invokes GETIO to read and check the next block. It then returns to the calling subroutine.

LOGIC OF MACRO SERVICES

The logic of the GET, PUT, and PUTX macro instructions can be broken down into three phases. Phase 1 deals with the communication between the problem program and the body of the QSAM routine. Phase 2 describes the initialization which is done only for: the first of these macro instructions issued on a data set; the first issued after a SETL type-E or -B has repositioned the data set; or the first after FEOV has advanced to a new volume in the data set. Phase 3 will describe the functions performed for all the above macro instructions when subsequently issued.

GET Macro Processing

Phase 1 - Communication: When the user issues a GET macro instruction, the macro expansion sets a locate or move mode code in the DCB, and then establishes type-1 linkage to the QSAM entry point whose V-con is found in DCBGTV. At the entry point, the user's registers are saved, and base registers for QSAM are established. If CZCSAA is entered, phase 2 is begun by giving control to INITIO. If CZCSAG is entered, phase 3 is begun by giving control to GET.

Phase 2 - First GET; Initialization: INITIO fills in two DECBs, for all data sets, and places their addresses in the DCB. It sets the forward or backward reads byte in the DECB, and sets the addresses of the buffers obtained by SAM Open into the data area pointers of the DECBs. Then the DCB address and the maximum block size are set into the DECBs. If single buffering is being done, the DECB pointers in the DCB are set equal to each other, and GETIO is invoked to read and check the first block. If double buffering is being done, Read/Write is invoked to read the first block, and GETIO is then invoked to read the second and check the read of the first.

Before returning to INITIO, GETIO gives control to COMIO to initialize buffer addresses. COMIO calculates the actual size of the block read in, by subtracting the residual count in the CSW from the maximum block size. It sets the current record address in the DCB from the data area address in the DECB, and then calculates the end-of-buffer address using the actual block size. If the record format is variable, four is added to the record address to allow for the four system control bytes in front of the block, and the sum of the lengths of the records is checked against the block size. If they are not equal, and if the user had not specified an EROPT parameter of ACC in the DCB, the task abnormally terminates by executing an ABEND macro instruction. Otherwise, COMIO returns to GETIO, which immediately returns to INITIO. Before returning to CZCSAA, INITIO places the V-con of CZCSAG in DCBGTV and DCBGTR, in order that the next GET issued by the user will enter QSAM at that point.

After INITIO has returned control to CZCSAA, the Get subroutine is invoked, and phase 3 is entered.

Phase 3 - Operation: If processing on the current buffer is complete, Get invokes either PUTXIO, to write an updated block back to an update data set, and read in the next block, or it invokes GETIO to refill the completed buffer. After the return from either PUTXIO or GETIO, or if processing on the current buffer is not yet complete, GET calculates the current record address, and if the GET macro instruction was in locate mode, sets the current record address into register 1 in the user's save area. If the GET macro instruction was in move mode, the record is moved into the user's work area.

Then the record address is incremented, by the length of the current logical record, so as to point to the end of the record. If it is within the block, an immediate return is made to the entry section. If it is at the end of the block, a flag in the DCB is set on to indicate that processing on the current buffer is complete, and the return is made. If it lies outside the block, and if the user has not specified an EROPT parameter of ACC in the DCB, the task abnormally terminates by executing an ABEND macro instruction. Otherwise, the return to the entry section is made.

Since processing on any GET macro instruction is now complete, both CZCSAA and CZCSAG issue a RETURN macro instruction, to restore the user's registers, and link back to the problem program.

PUT Macro Processing

Phase 1 - Communication: When the user issues a PUT macro instruction, the macro expansion indicates either locate or move mode in the DCB, and establishes type-1 linkage to the QSAM entry point whose V-con is in DCBPTV. At the entry point, the user's registers are saved, and base registers for QSAM are established. If CZCSAB is entered, phase 2 is begun by giving control to INITIO. If CZCSAW is entered, phase 3 is begun by giving control to PUT.

Phase 2 - First PUT - Initialization: INITIO builds two DECBs for all output data sets, as described in phase 2 of the GET macro instruction, except that it sets the DECB type code to indicate that only writes are to be done. It then invokes the BSAM Note routine (CZCRN) to obtain the relative address within the volume (TTRZ or ZZCC) of the last block read or written. If single buffering is being done, the DECB pointers in the DCB are set equal to each other. If double buffering is being done, the completion code in the DECB is set to indicate "complete with no errors", so that the first check, which will be done on an unused DECB, will return normally. Following this, if the record format is variable, the two length control bytes at the beginning of the block are initially set to four, and the current logical record address is set to the beginning-of-buffer address, plus four. The end-of-buffer address is calculated by adding the maximum block size to the beginning-of-buffer address.

The V-con of CZCSAW is placed in DCBPTV so that the next PUT issued will enter QSAM at that point, and a return is made to CZCSAB. From CZCSAB, control is then passed to Put, and phase 3 is entered.

Phase 3 - Operation: If the current block of records is complete, Put gives control to PUTIO to write the block out and check the previous write operation. The record format is checked, since variable format records are treated separately from fixed and undefined, the end-of-buffer address is set to the current record address plus the logical record length. Then, for both fixed and undefined records, if the PUT macro instruction is in locate mode, the current record address is set into register 1 in the user's save area. If the PUT is in move mode, the record is moved from the user's work area to the current record address, and in order to support substitute mode exchange buffering, the address of the user's work area is set into register 1 in his register save area.

If the block is complete, the End of Buffer flag in the DCB is set on. If the block is not yet complete, or after the End of Buffer flag is set, the logical record count for the current block is increased by one, and a return is made to the entry section. If the record overflowed the buffer, the task abnormally terminates by executing an ABEND macro instruction.

For variable format records, a check is first made to see if the last PUT issued was in locate mode. If so, the length control bytes of the record which was subsequently built in the buffer are checked to be sure that the record is not larger than the length previously estimated by the user. If it is larger, and if the record is too long to fit into the buffer, the task is abnormally terminated.

Following this, or if the last PUT was not in locate mode, if the current record will not fit into the buffer, PUTIO is invoked to write out the buffer, check the previous write, and provide a new buffer address. After return from PUTIO, or if the current record will fit into the buffer, the current record address is set into register 1 of the user's register save area if the PUT is in locate mode, and a return is made to the entry section. If the PUT is in move mode, the record is moved from

the user's work area to the current record address, which is then incremented by the length of the record.

Since processing on any PUT macro instruction is then complete, both CZCSAB and CZCSAQ issue a RETURN macro instruction to restore the user's registers and return to the problem program.

PUTX Macro Processing

Phase 1 - Communication: When a PUTX macro instruction is issued, the macro expansion establishes type-1 linkage to CZCSAX, whose V-con is in DCBPXV. At the entry point, the user's registers are saved, and base registers for QSAM are established. If the PUTX is an update PUTX, it must have been preceded by a locate mode GET on the same data set, and therefore cannot be the first macro issued. Hence, phase 3 is begun by giving control to the PUTX subroutine. If it is an output PUTX, a check is made to see if it is the first macro issued on the data set, and if so, phase 2 is begun by giving control to INITIO. Otherwise, PUTX is invoked.

Phase 2 - First PUTX - Initialization: The initialization for the first PUTX on a data set is accomplished by INITIO in exactly the same manner as that for the first PUT on a data set. When initialization is complete, a return is made to CZCSAX, which then gives control to PUTX.

Phase 3 - Operation: If the data set on which the PUTX macro instruction was issued is opened for Output, the associated data set must be opened for Output, and if it is the task abnormally terminates by executing an ABEND macro instruction. The record which is to be put out must have been retrieved by a locate-mode GET on the associated data set. If it was not, the task is abnormally terminated. Otherwise, a flag is set in the output DCB to indicate that a move-mode PUT is to be performed, the logical address field of the input DCB is supplied as the address of the record to be output, and control is given to the PUT subroutine (whose operation was discussed in phase 3 of the PUT macro instruction).

The data set must be opened for Output and Update, and the last logical record must have been retrieved by a locate mode GET; otherwise, abnormal termination occurs.

A flag is set in the DCB to indicate that a PUTX has been issued on the current block. If processing on the current block is complete, PUTXIO is invoked to write the updated block back to the data set, and to read in the next block if called by the GET subroutine. After the return from PUTXIO, or if processing on the block was not complete, a return is made to CZCSAX, which then issues a RETURN macro instruction to restore the user's registers and link back to the problem program.

TRUNC Macro Processing

The macro instructions TRUNC and RELSE require no initialization phase, since they perform no functions if they are the first macro instructions issued on a data set. Therefore, they will be discussed in only two phases, communication and operation.

Phase - Communication: The expansion of the TRUNC macro instruction establishes type-1 linkage to CZCSAT, whose V-con is defined within the expansion. The entry section saves the user's registers, establishes base registers for QSAM, and gives control to the TRUNC subroutine.

Phase 2 - Operation: TRUNC makes an immediate return to the entry section under the following conditions:

1. If processing on the current block is already complete, or has not yet begun;
2. If the record format is undefined;
3. If the data set is opened for neither Output nor Update;
4. If no GET or PUT has previously been issued on the data set.

Otherwise, if the data set is opened for UPDATE, TRUNC sets the End of Buffer flag in the DCB so that the next GET will retrieve the first logical record of the following block. If the data set is opened for Output, and if the record format is variable, and if the last PUT was in locate mode, the last logical record is checked to be certain it does not overflow the buffer. If it does, the task is abnormally terminated. In all other cases, the actual block size is calculated, and PUTIO is invoked to write out the block.

RELSE Macro Processing

Phase 1 - Communication: The expansion of the RELSE macro instruction establishes type-1 linkage to CZCSAR, whose V-con is defined within the expansion. The entry section saves the user's registers, establishes base registers for QSAM, and gives control to the RELSE subroutine.

Phase 2 - Operation: RELSE makes an immediate return to the entry section if processing has not yet begun on the current block, or if the data set is opened for Output. Otherwise, it sets the End of

buffer flag in the DCB so that the next GET on the data set will retrieve the first logical record of the following block. Then RELSE returns to CZCSAR, which issues a RETURN macro instruction to restore the user's registers and link back to the problem program.

SETL Macro Processing

The communication phase is the same for all types of SETL macro instructions, but the operation phases must be discussed separately. The expansion of each SETL macro instruction sets a code in the DCB to indicate its type (C, R, P, E or B), and establishes type-1 linkage to CZCSAS, whose V-con is found in DCBSLV. The entry section saves the user's registers and establishes base registers for QSAM. It then gives control to SETLC, SETLR, SETLP, or SETLEB. The routine ABENDs if SETL was not requested in the DCB MACRF. Single buffering is done for SETL.

SETL Type-C (Current Block): The SETLC subroutine first invokes CHECK to check the last read or write performed, and then gives control to BSAM Note (CZCRN) to establish the relative address (TTRZ or ZZCC) within the data set of the last block read or written. Since that may not be the block currently being processed, it must be determined whether or not additional spacing will be needed. The data set is repositioned to that retrieval address.

If the data set is opened for Output, and if it is not positioned to the beginning, 1 is added to the retrieval address, since the current block will always be 1 beyond the last block written. The count of logical records already processed within the current block is then returned to the user, along with the retrieval address.

If the data set is opened for Update, a test must be made to see if the current record address points to the beginning of the block. This is possible only when a PUTX macro instruction, issued on the last record of a block, has caused that block to be written back to the data set and a new block to be read in. In this case, a backspace will be required to retrieve the block within which the desired record resides. Therefore, the count of records within the last block, with the high order bit set on to indicate that a backspace is needed, is returned to the user with the retrieval address. If the current record address points within the block, no backspace is needed, since a GET must already have been issued on the block. Hence, the count of records already processed within the current block is returned to the user with the retrieval address.

Since a data set opened for Input or RDBACK employs double buffering, one block beyond the current one being processed will be the one marked by CZCRN. Therefore, either a backspace or a forward space will always be required to retrieve the current block. The count of records processed within the current block (with the high order bit set on to indicate that a backward or forward space is needed) is returned to the user with the relative address obtained by CZCRN.

When processing is complete, a return is made to CZCSAS, which then issues a RETURN macro instruction to restore the user's registers and link back to the problem program.

SETL Type-R (Retrieval Address): The SETLR subroutine first invokes TREOV to clear any outstanding I/O requests. Then, using the retrieval address provided by the user as a parameter, the SETLR subroutine invokes POINT to reposition the data set to the block specified by the retrieval address. If a backward or forward space is required (see SETL type-C for the manner in which this is determined), either Backspace is invoked to backspace one block, or Control is invoked to forward space one block.

Following this, the original open option is saved, and if the data set is opened for Output, the open option is set to indicate that the block to which the data set is now positioned must be read back in. INITIO is invoked to set up new DECBs and initiate a read of the desired block. INITIO will function as it does in phase 2 of the GET macro instruction, except in the case of an output data set, when it sets the DECB type code to "read" and invokes Read/Write to read in the desired block and GETIO to check it.

When INITIO returns control to SETLR, the current record address points to the first record of the block. The record count provided by the user is then decreased by one, and if the result is not zero, the Get subroutine is invoked to calculate the address of the next logical record. Again the record count is decreased by one, and GET is invoked if it is not zero. When the record count reaches zero, the current record address points to the desired record. At this point, the original open option is restored, and, if the data set is opened for Output, the DECB type code is reset to "write". A return is made to CZCSAS, which issues a RETURN macro instruction to restore the user's registers and link back to the problem program.

SETL Type-P (Previous Record): The SETLP subroutine first checks to see if processing has begun on the current buffer. If

so, the previous record must be within the current block. Its address is calculated, the record count within the block is decreased by 1, and a return is made to CZCSAS.

If the current buffer is empty, however, the previous record lies within the last block processed. In this case, TREOV is invoked to clear any outstanding I/O requests. Upon return from TREOV, if the Write Request flag in the DCB is on, an immediate backspace of one block is made. If the data set is opened for RDBACK, Control is invoked to space forward one block. If the data set is opened for Input, Check is invoked to check the last read and then Backspace is invoked to backspace the data set three times, since it is positioned at the end of the third block beyond the one in which the desired record lies. If the data set is opened for Output, it is positioned at the end of the block containing the desired record, and BSP is therefore invoked to backspace one block. If any positioning errors occur, SYNAD is invoked.

When the data set is correctly positioned, the open option is saved, and if the data set is opened for Output, the open option is temporarily set to Input. INITIO is invoked to build new DECBs and initiate a read of the desired block. Then GET is invoked to calculate the next record address until the End of Buffer flag is on, at which time the data set is opened for Output, the original open option is restored, the DECB type code is reset to "write", and Flush is invoked to purge outstanding read requests. Finally, for all OPEN options, the End of Buffer flag is set off, and a return is made to CZCSAS, which issues a RETURN macro instruction to restore the user's registers and link back to the problem program.

SETL Type-E or -B (End or Beginning): The SETLEB subroutine makes an immediate return to CZCSAS, if the data set is opened for Output and type-E is specified. Otherwise, TREOV is invoked to clear all outstanding write requests. Then, if the device is direct access, register 0 is set to zero for type-B, and Point is invoked to position the data set at the beginning or the end.

If the device employs magnetic tape, and the data set is opened for Rdback, and type-B is specified, it must be spaced backward to the end. If the data set is opened for Input, it must be spaced backward to the beginning for type-B, or spaced forward to the end for type-E. Control is invoked to perform the spacing of a data set on magnetic tape.

When the data set is correctly positioned, the Get and Put V-cons in DCBGTV and DCBPTV, respectively, are set to CZCSAA and CZCSAB so that the next GET or PUT on the data set. Then a return is made to CZCSAS, which issues a RETURN macro instruction to restore the user's registers and link back to the problem program.

CLOSE and FEOV Functions Performed by QSAM

During processing of a CLOSE or FEOV macro instructions, the TREOV subroutine of (QSAM) is used to perform those functions necessary to closing out a volume or a data set, such as writing out the last buffer of an output data set, or purging any read requests which may have been issued on an input data set after the CLOSE or FEOV was issued.

When it is being invoked by SAM Close or FEOV, TREOV is given control by the QSAM entry section CZCSAV. SAM Close or FEOV establishes type-1 linkage to CZCSAV, where base registers for QSAM are established and control is passed to TREOV. (Note that TREOV is also used internally by other subroutines of QSAM during the processing of QSAM macro instructions, and returns to whatever routine called it.)

If the data set has been opened for RDBACK or Input, the Flush subroutine of TREOV is invoked to purge outstanding I/O requests, and upon return from Flush, the first Get or Put V-cons are moved into the DCB, and an immediate return is made to CZCSAV, or to the calling subroutine.

If the data set is opened for Output and the current buffer is empty, the last block of the data set has already been written out, and, if it has not yet been checked, the Check subroutine is invoked to perform the check. Otherwise, or upon return from Check, a return is made to the entry routine or to the calling subroutine. If the buffer is not empty, it must be written out as the last block of the data set. If the last PUT issued was in locate mode, and if the records are variable format, a check is made to be certain that the last record placed in the buffer does not overflow the end of the buffer. If it does, the task is abnormally terminated. Otherwise, the actual size of the block is calculated and PUTIO is invoked to write out the current buffer and check the previous write. (If single buffering is being done, PUTIO will write and check the same buffer.) Upon return from PUTIO, if any unchecked DECBs remain, Check is invoked to perform the check. Then a return is made to CZCSAV or to the calling subroutine.

If the data set is opened for Update, and if no PUTX macro instruction has been

issued on the current block, TREOV gives control to Flush to purge any outstanding I/O requests, and then returns to the entry section or the calling subroutine. If a PUTX has been issued on the current block, however, the Read/Write subroutine is invoked to write the updated block back to the data set, and the Check subroutine is invoked to check on the completion of the write request. Then the return is made to the entry section or the calling subroutine.

PART IV

RTAM/MTT ACCESS METHODS SUPPORT

For the MTT user, there is a virtual storage routine (CZCTC) which performs functions analogous to access methods Read, Write, Find, or Close routines. The MTT command, through the command system, is processed by this routine, and the CLEARQ and FREEQ macro instructions which clear pending work for a terminal and logically disconnect a terminal respectively.

Terminal Task Control Routine (CZCTC)

The Terminal Task Control routine provides an interface with the Terminal Communications Subprocessor in the resident supervisor. None of the macro instructions entered by the MTT user issue their own I/O, such as Read or Write. The method used by CZCTC is to set information into the application task's terminal control table (TCT), and then to issue the ATCS macro instruction which calls the Terminal Control Subprocessor to perform the requested operation.

When processing the MTT enable, communication with the user is effected through PRMPT (CZATJ) in order to prompt the user for unentered or unacceptable parameters.

The processors described are for the MTT command, and the FINDQ, READQ, WRITEQ, CLEARQ, and FREEQ macro instructions. Each has a separate entry point in CZCTC (Chart RA).

The subroutine, CZCTC7, tests the validity of the device (line) number for each of the macro instructions (Chart RA).

MTT Enable

Entry is the result of the MTT command given from a terminal, and passed through the command system. The application program is loaded, parameters are checked, the schedule table level is entered, and the virtual memory necessary for the task is reserved. When the application program is complete it is unloaded, the original schedule table level restored, and the virtual storage released. (The application program in this case is the control program for the multiterminal task operation (Chart RA).)

Entry Point: CZCTC1 -- From the command system via type-1 linkage.

Input: Register 1 contains the address of a 4-word parameter list:

Word 1 -- Address of a program name which is a maximum of 8-characters long. The byte preceding this address is a count of the characters in the name.

Word 2 -- Address of 4-byte count of the maximum number of terminals which may be simultaneously connected to the task (1-4095).

Word 3 -- Address of 3-byte schedule table level (1-255).

Word 4 -- Address of 4-byte count of the size of the input buffer associated with each terminal line (16-4076), default 200.

Modules Called: Via type-2 linkage, unless specified:

PRMPT (CZATJ1) -- Prompt user for parameters, inform user of error.

GETMAIN (CZCGA2) -- Allocate necessary virtual storage.

ABEND (CZACP1) -- Abnormal task termination.

FREEMAIN (CZCGA3) -- Free virtual storage.

There is also a type-1 call to the application program named in the MTT command.

Exits:
Normal -- Return to the calling routine.

Error --

- Return to the calling routine after informing SYSOUT that the user is not authorized to issue MTT.
- Type-2 exit to SYSER followed by a return to the calling routine, if the return from a PRMPT call is an error code.
- SYSER followed by ABEND, if a non-zero return code is received after a FREEQ ALL macro is issued.

Operation: This text is keyed to the flowchart for CZCTC (Chart RA) and is referenced by label.

The authority of the user issuing the MTT is tested and the application program is loaded or prompted for. The user must have an 'O' or 'P' authority (A00-A00X-A01-A02).

The parameters are tested, and prompted for if necessary (A03-A17-A14B).

The number of pages required by the task is computed and allocated, based on the number of application buffer pages needed plus the number of TCT pages necessary (A18-A18A).

The CONN SVC is issued which transfers control to CEAR4 to perform initialization in real-core. The multiterminal system control block (MTSCB) is built and the TCT and buffer page allocated. If the task is already MTT, it is returned to normal status and the CONN is reissued (TC1A-TC1B-TC1B1).

The current schedule table level is saved and the new level entered (TC1A-TC1C).

The application program, which is the MTT control program, is now dispatched via a type-1 call.

When the MTT application returns, it is unloaded (DELETE) and FREEQ ALL is issued to logically disconnect the terminals. The DCON macro cleans up main storage, the original schedule table level is returned, and the virtual storage is freed.

A return is made to the command system.

FINDQ Macro

FINDQ tests a specific line number, or polls the MTT application program's work queue (TCT) to find a terminal with work to be done. A return code specifying the work to be done, or that there is no work to do, is returned to the calling program.

Entry Point: CZCTC2 -- Via type-1 linkage.

Input: Register 1 contains the address of CHAFNQ. FNQCTL (in CHAFQN) is a 2-byte field:

FFFF -- Polling operation.

XXXX -- Unique device number from 0000 to a user-specified maximum. The system maximum is 4095.

Modules Called: None.

Exits: Return to the calling routine with one of the following codes in register 15:

Code	Condition
'00'	No work.
'04'	Invalid relative line number (not connected to application).
'08'	Initial connection of device.
'0C'	Attention from terminal.
'10'	Solid I/O error on terminal line.
'14'	Message Out complete (from previous WRITEQ).
'18'	Message In complete (from previous WRITEQ/RESP).
'1C'	Negative response from input component.
'20'	Message In overflowed buffer.

Operation: If polling is specified, the TCT slots are scanned for work starting with the slot after the halt of the previous scan. The result is returned as a code in register 15. (See 'Exits.')

CHAFNQ fields are set with the device type and symbolic device address.

Message In also sets the length of the message and its address in CHAFNQ.

If a specific line is to be tested, no polling takes place if there is no work in its TCT.

A SYSER is issued if there is work in the TCT, but it cannot be identified. The SYSER is followed by a return to the user with a return code of X'04'.

READQ Macro

READQ posts the read request and associated options in the TCT slot and passes control to the resident supervisor via ATCS. For any return code other than '00', the read operation has not been initiated.

Entry Point: CZCTC3 -- Via type-1 linkage.

Input: Register 1 contains the address of CHARDQ. RDQDEV (in CHARDQ) is a 2-byte device (line) number.

Modules Called: None.

Exits: Return with a code in register 15:

Code	Condition
'00'	Normal completion.
'04'	Device number invalid, component select invalid.
'08'	Previous operation incomplete.
'0C'	Attention from terminal.
'10'	Unrecoverable I/O error on terminal line.

Operation: The text is keyed to chart RA, Entry CZCTC3. References are by label.

The device (line) number, and the Attention and Previous Operation flags in the TCT are tested (Entry CZCTC3-TC3A).

Parameter fields are tested and set accordingly (TC3B-TC3G). The Interruption bit specifies that an application program will process external interruptions generated upon completion of a read operation. The interruptions will be ignored if it is not set.

The Component Select field specifies a type of 1050 unit:

0 Any input component

5 Terminal keyboard

6 Reader 1

7 Reader 2

The Terminal Communications Subprocessor is invoked to perform the operation via ATCS, and upon completion the result is tested, and an appropriate code returned (TC3G-TC3J).

WRITEQ Macro

WRITEQ posts the write requests and associated options in the application TCT slot and passes control to the resident supervisor via ATCS for execution. For any return code other than '00', the write operation has not been initiated.

Entry Point: CZCTC4 -- Via type-1 linkage.

Input: Register 1 contains the address of CHAWRQ. WRQDEV (in CHAWRQ) is a 2-byte device (line) number.

Modules Called: None.

Exits: Return to the calling routine with a code in register 15:

Code	Condition
'00'	Normal return.
'04'	Invalid relative line number, Component-Out field.
'08'	Busy, I/O outstanding, HIO not complete.
'0C'	Attention interruption from terminal.
'10'	Solid I/O error on line.
'14'	Message length not 1-4080 bytes.

Operation: The text is keyed to the flowchart (Chart RA, Entry CZCTC4) by label.

The device (line) number, 'break' option, outstanding I/O, and 'busy' options are tested (Entry CZCTC4-TC43A).

Parameter options are set in TCT (TC44-TC4C).

The data is forced into core, and the ATCS macro issued to execute the write operation. The results are tested upon return from the resident supervisor, and a code set accordingly (TC4D-TC4C3).

CLEARQ Macro

The status byte of the application TCT slot is tested for work indications, and unless a READQ or WRITEQ is in progress, the contents of the TCT slot are saved and the byte is set to zero.

Entry Point: CZCTC5 -- Via type-1 linkage.

Input: Register 1 contains the address of CHACLQ. CLQDEV (in CHACLQ) is a 2-byte device (line) number.

Modules Called: None.

Exits: Return to the calling program with a code in register 15:

Code	Condition
'00'	Normal return.
'04'	Invalid device (line) number.
'08'	Busy, previous READQ or WRITEQ not completed.
'0C'	Attention from line.

Operation: The device number and 'busy' indicator are checked.

The work byte is saved, and the reset byte also. If a buffer is connected, ATCS is issued to perform the clear function (see Chart RA, labels TC51-TC5A1).

A return is made with the appropriate code.

FREEQ Macro

FREEQ will logically disconnect a specific terminal, or all terminals associated with an application program. If specified, a message is written out to the terminal before disconnecting. A physical disconnect may be requested, whereby the line to the device will be disabled -- this is handled in the resident supervisor.

Entry Point: CZCTC6 -- Via type-1 linkage.

Input: Register 1 contains the address of CHAFRQ. FRQDEV (in CHAFMQ) is a 2-byte device (line) number.

Modules Called: None.

Exits: Return to the calling program with a code in register 15:

Code	Condition
'00'	Normal return.
'04'	Invalid device (line) number.
'08'	Disconnect specified not 'logical' and not 'physical'.
'0C'	Message specified with zero length.
'10'	Message address illegal.

Operation: If the ALL option is set and the disconnect valid (physical or logical), the message is tested and ATCS issued to perform the request.

If a specific line is to be freed, the Free Function flag is set and the work byte cleared, before ATCS is issued. (See Chart RA, Entry CZCTC6.)

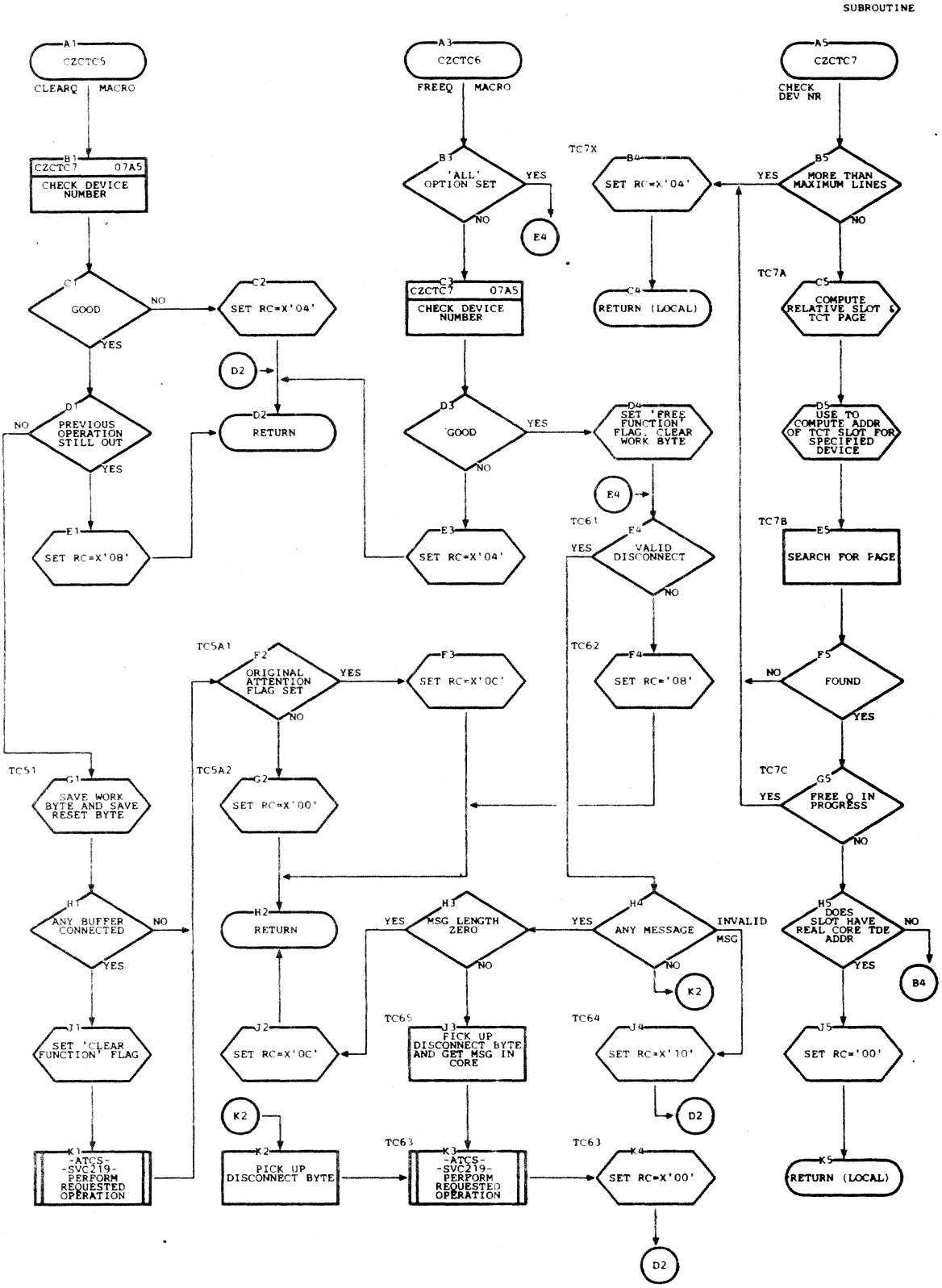
Program Logic Manual

GY28-2016-5

Access Methods

Flowcharts on pages 199-438 were not scanned.

Chart RA. Terminal Task Control - CZCTC (Page 7 of 7)



CHAADC-Explicit adcon group
  used by:
    CZCTC-Terminal Task Control

CHABCT-BULKOMM Table
  used by:
    CZCQK-VDMEP

CHABPL-Buffer Page List
  used by:
    CZCMA-GETBUF
    CZCNA-FREEBUF
    CZCNB-FREEPOOL

CHADBP-DEB Page
  used by:
    CZCMC-MSAM Open
    CZCMD-SETUR
    CZCME-DOMSAM
    CZCMG-MSAM Posting and Error Retry
    CZCMH-MSAM Finish
    CZCMI-MSAM Close

CHACLQ-CLEARQ
  used by:
    CZCTC-Terminal Task Control

CHADCB-Data Control Block
  used by:
    CZCLA-Open Common
    CZCLB-Close Common
    CZCLD-FEOV
    CZCMA-GETBUF
    CZCMB-GETPOOL
    CZCMC-MSAM Open
    CZCMD-SETUR
    CZCME-DOMSAM
    CZCMF-MSAM Read/Write
    CZCMG-MSAM Posting and Error Retry
    CZCMH-MSAM Finish
    CZCMI-MSAM Close
    CZCNA-FREEBUF
    CZCNB-FREEPOOL
    CZCOA-OPENVAM
    CZCOB-CLOSEVAM
    CZCOC-MOVEPAGE
    CZCOD-Insert/Delete Page
    CZCOE-REQPAGE
    CZCOF-Insert
    CZCOG-Reclaim
    CZCOJ-Find
    CZCOK-Stow
    CZCOL-Search
    CZCOM-Extend POD
    CZCON-Relocate Members
    CZCOD-GETNUMBR
    CZCOP-VSAM Open
    CZCOQ-VSAM Close
    CZCOR-VSAM Get
    CZCOS-VSAM Put
    CZCOT-SETL
    CZCOU-PUTX
    CZCOV-FLUSHBUF
    CZCPA-VISAM Put
    CZCPB-VISAM Get
    CZCPC-SETL
    CZCPE-Read/Write
    CZCPI-GETPAGE
    CZCPL-Add Directory Entry
    CZCPZ-VISAM Open
    CZCQA-VISAM Close
    CZCQE-Search SDST
    CZCQI-Expand RESTBL
    CZCQQ-VAM ABEND Interlock Release
    CZCRA-BSAM Read/Write
    CZCRB-Control
    CZCRC-Check
    CZCRG-Backspace
    CZCRM-Point
    CZCRN-Note
    CZCRP-SAM Posting and Error Retry
    CZCSA-QSAM
    CZCSB-IOREQ
    CZCSC-IOR Open
    CZCSD-IOR Close
    CZCSE-IOREQ Posting
    CZCWA-ASCII Translation and Conversion
    CZCWB-Build Common DEB
    CZCWC-SAM Close
    CZCWD-DAOPEN
    CZCWL-Build DA DEB
    CZCWO-SAM Open Mainline
    CZCWT-Tape Open
    CZCWY-Tape Data Set Label
    CZCXD-DA Output EOV
    CZCXE-Mainline EOV
    CZCXI-DA Input EOV
    CZCXN-DA Input LABEL
    CZCXO-Tape Output EOV
    CZCXS-Set DSCB
    CZCXT-Tape Input EOV
    CZCXU-DA Output Label
    CZCXX-Concatenation
    CZCYA-TAM Open
    CZCYG-TAM Close
    CZCYM-TAM Read/Write
    CZCZA-TAM Posting

CHADEB-Data Extent Block
  used by:
    CZCLA-Open Common
    CZCLB-Close Common
    CZCLD-Force End of Volume
    CZCMA-GETBUF
    CZCMB-GETPOOL
    CZCMC-MSAM Open
    CZCMD-SETUR
    CZCME-DOMSAM
    CZCMF-MSAM Read/Write
    CZCMG-MSAM Posting and Error Retry
    CZCMH-MSAM Finish
    CZCMI-MSAM Close
    CZCNA-FREEBUF
    CZCNB-FREEPOOL
    CZCRA-BSAM Read/Write
    CZCRB-Control
    CZCRC-Check
    CZCRG-Backspace

CZCRM-Point
CZCRN-Note
CZCRP-SAM Posting and Error Retry
CZCSA-QSAM
CZCSB-IOREQ
CZCSC-IOR Open
CZCSD-IOR Close
CZCSE-IOREQ Posting
CZCWB-Build Common DEB
CZCWC-SAM Close
CZCWD-DAOPEN
CZCWL-Build DA DEB
CZCWM-Message Writer
CZCWO-SAM Open Mainline
CZCWP-Tape Positioning
CZCWR-Read Format-3 DSCB
CZCWT-Tape Open
CZCWY-Tape Data Set Label
CZCXD-DA Output EOV
CZCXE-Mainline EOV
CZCXI-DA Input EOV
CZCXN-DA Input Label
CZCXS-Set DSCB
CZCXT-Tape Input EOV
CZCXU-DA Output Label
CZCXX-Concatenation
CZCYA-TAM Open
CZCYG-TAM Close
CZCYM-TAM Read/Write
CZCZA-TAM Posting

CHADEC-Data Event Control Block
used by:
CZCMC-MSAM Open
CZCME-DOMSAM
CZCMF-MSAM Read/Write
CZCMG-MSAM Posting and Error Retry
CZCMH-MSAM Finish
CZCMI-MSAM Close
CZCPE-Read/Write
CZCRA-BSAM Read/Write
CZCRB-Control
CZCRC-Check
CZCRP-SAM Posting and Error Retry
CZCSA-QSAM
CZCSB-IOREQ
CZCSD-IOR Close
CZCSE-IOREQ Posting
CZCWC-SAM Close
CZCWO-SAM Open Mainline
CZCWY-Tape Data Set Label
CZCXE-Mainline EOV
CZCXS-Set DSCB
CZCYG-TAM Close
CZCYM-TAM Read/Write
CZCZA-TAM Posting

CHADHD-DCB Header
used by:
CZCLB-Close Common
CZCOA-OPENVAM
CZCOB-CLOSEVAM
CZCOC-MOVEPAGE
CZCOD-Insert/Delete Page
CZCOE-REQPAGE
CZCOF-Insert
CZCOG-Reclaim
CZCOJ-Find
CZCOK-Stow
CZCOL-Search
CZCOM-Extend POD
CZCON-Relocate Members
CZCOO-GETNUMBR
CZCOP-VSAM Open
CZCOQ-VSAM Close
CZCOR-VSAM Get
CZCOS-VSAM Put
CZCOT-SETL
CZCOU-PUTX
CZCOV-FLUSHBUF
CZCPA-VISAM Put
CZCPB-VISAM Get
CZCPC-SETL
CZCPE-Read/Write
CZCPI-GETPAGE
CZCPL-Add Directory Entry
CZCPZ-VISAM Open
CZCQA-VISAM Close
CZCQI-Expand RESTBL
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release

CHADSC-DSCB Format-1 & -3
used by:
CZCWD-DAOPEN
CZCWL-Build DA DEB
CZCWO-SAM Open Mainline
CZCWR-Read Format-3 DSCB
CZCXD-DA Output EOV
CZCXI-DA Input EOV
CZCXS-Set DSCB

CHADSV-DSCB Format-A & -B
used by:
CZCOA-OPENVAM
CZCOB-CLOSEVAM

CHAEPE-RESTBL External Page Entry
used by:
CZCOC-MOVEPAGE
CZCOE-REQPAGE
CZCOF-Insert
CZCOG-Reclaim
CZCOO-GETNUMBR

CHAFNQ-FINDQ
used by:
CZCTC-Terminal Task Control

CHAFRQ-FREEQ
used by:
CZCTC-Terminal Task Control

CHAGSM-General Services Macro Table
used by:
CZCLA-Open Common
CZCLB-Close Common

CHAICB-Interruption Control Block
used by:
CZCMC-MSAM Open
CZCMD-SETUR
CZCMG-MSAM Posting and Error Retry
CZCMI-MSAM Close

CHAIOR-I/O Request Control Block
used by:
CZCMC-MSAM Open
CZCMD-SETUR
CZCMF-MSAM Read/Write

CZCMG-MSAM Posting and Error Retry
CZCMH-MSAM Finish
CZCRA-BSAM Read/Write
CZCRB-Control
CZCRP-SAM Posting and Error Retry
CZCSB-IOREQ
CZCSE-IOREQ Posting
CZCYM-TAM Read/Write
CZCZA-TAM Posting

CHAISA-Interruption Storage Area
used by:
CZCEI-VMIER
CZCLA-Open Common
CZCMA-GETBUF
CZCMG-GETPOOL
CZCMD-SETUR
CZCME-DOMSAM
CZCMF-MSAM Read/Write
CZCMG-MSAM Posting and Error Retry
CZCMH-MSAM Finish
CZCNA-FREEBUF
CZCNB-FREEPOOL
CZCPA-VISAM Put
CZCPB-VISAM Get
CZCPC-SETL
CZCQE-Search SDST
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release
CZCRP-SAM Posting and Error Retry
CZCSA-QSAM
CZCSB-IOREQ
CZCSE-IOREQ Posting
CZCWD-DAOPEN
CZCWO-SAM Open/Mainline
CZCWT-Tape Open
CZCWY-Tape Data Set Label
CZCXE-Mainline EOV
CZCXN-DA Input Label
CZCYG-TAM Close
CZCZA-TAM Posting

CHALB1-Data Set Header/Trailer Label 1
used by:
CZCWM-Message Writer
CZCWY-Tape Data Set Label

CHALB2-Data Set Header/Trailer Label 2
used by:
CZCWY-Tape Data Set Label

CHAMHD-RESTBL Member Header
used by:
CZCOA-OPENVAM
CZCOB-CLOSEVAM
CZCOC-MOVEPAGE
CZCOE-REQPAGE
CZCOJ-Find
CZCOK-Stow
CZCON-Relocate Members
CZCOO-GETNUMBR
CZCQI-Expand RESTBL
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release

CHAPOD-Partitioned Organization Directory
used by:
CZCOJ-Find
CZCOK-Stow
CZCOL-Search
CZCOM-Extend POD
CZCON-Relocate Members
CZCOO-GETNUMBR

CHAPOE-Directory Alias Descriptor
used by:
CZCOJ-Find
CZCOK-Stow
CZCOL-Search

CHAPOM-Directory Member Descriptor
used by:
CZCOJ-Find
CZCOK-Stow
CZCOL-Search
CZCON-Relocate Members
CZCOO-GETNUMBR

CHAPVT-Public/Private Volume Table
used by:
CZCEI-VMIER

CHARDQ-READQ
used by:
CZCTC-Terminal Task Control

CHARHD-RESTBL Header
used by:
CZCOA-OPENVAM
CZCOB-CLOSEVAM
CZCOC-MOVEPAGE
CZCOD-Insert/Delete Page
CZCOE-REQPAGE
CZCOF-Insert
CZCOG-Reclaim
CZCOJ-Find
CZCOK-Stow
CZCOM-Extend POD
CZCON-Relocate Members
CZCOO-GETNUMBR
CZCOP-OPENSEQ
CZCOQ-CLOSESEQ
CZCOR-VSAM Get
CZCOS-VSAM Put
CZCOT-SETL
CZCOU-PUTX
CZCPA-VISAM Put
CZCPB-VISAM GET
CZCPC-SETL
CZCPE-Read/Write
CZCPI-GETPAGE
CZCPL-Add Directory Entry
CZCPZ-VISAM Open
CZCQA-VISAM Close
CZCQI-Expand RESTBL
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release

CHASAR-System Activity and Resource Table
used by:
CZCTC-Terminal Task Control

CHASCB-SAM Communication Block
used by:
CZCWB-Build Common DEB
CZCWC-SAM Close
CZCWD-DAOPEN
CZCWL-Build DA DEB
CZCWM-Message Writer
CZCWO-SAM Open Mainline

CZCWP-Tape Positioning
CZCWR-Read Format-3 DSCB
CZCWT-Tape Open
CZCWV-Volume Sequence Convert
CZCWY-Tape Data Set Label
CZCXD-DA Output EOV
CZCXE-Mainline EOV
CZCXI-DA Input EOV
CZCXN-DA Input Label
CZCXO-Tape Output EOV
CZCXS-Set DSCB
CZCXT-Tape Input EOV
CZCXU-DA Output Label
CZCXX-Concatenation

CHASDA-Symbolic Device Allocation
used by:
CZCEI-VMIER
CZCMC-MSAM Open
CZCMD-SETUR
CZCMG-MSAM Posting and Error Retry
CZCMI-MSAM Close
CZCQK-VDMEP
CZCRA-BSAM Read/Write
CZCRB-Control
CZCRC-Check
CZCRG-Backspace
CZCRP-SAM Posting
CZCSC-IOR Open
CZCWB-Build Common DEB
CZCWC-SAM Close
CZCWD-DAOPEN
CZCWL-Build DA DEB
CZCWM-Message Writer
CZCWP-Tape Positioning
CZCWY-Tape Data Set Label
CZCXS-Set DSCB
CZCYA-TAM Open
CZCYG-TAM Close
CZCYM-TAM Read/Write
CZCZA-TAM Posting

CHASDE-Shared Data Set Entry
used by:
CZCOA-OPENVAM
CZCOB-CLOSEVAM
CZCQE-Search SDST
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release

CHASET-BULKIO S-Entry Table
used by:
CZCQK-VDMEP

CHASDM-Shared Data Set Member
used by:
CZCOA-OPENVAM
CZCOB-CLOSEVAM
CZCQE-Search SDST
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release

CHASDS-Shared Data Set Table
used by:
CZCOA-OPENVAM
CZCOB-CLOSEVAM
CZCQE-Search SDST
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release

CHASDT-I/O Statistical Data Table
used by:
CZCMG-MSAM Posting and Error Retry
CZCRP-SAM Posting and Error Retry

CHATCM-Task Common Table
used by:
CZCEI-VMIER
CZCQK-VDMEP
CZCTC-Terminal Task Control

CHATCT-Terminal Control Table
used by:
CZCTC-Terminal Task Control

CHATDT-Task Data Definition Table
used by:
CZCEI-VMIER
CZCLA-Open Common
CZCLB-Close Common
CZCMC-MSAM Open
CZCMI-MSAM Close
CZCOA-OPENVAM
CZCOB-CLOSEVAM
CZCOE-REQPAGE
CZCQE-Search SDST
CZCQI-Expand RESTBL
CZCQK-VDMEP
CZCQQ-VAM ABEND Interlock Release
CZCSC-IOR Open
CZCWB-Build Common DEB
CZCWC-SAM Close
CZCWD-DAOPEN
CZCWL-Build DA DEB
CZCWM-Message Writer
CZCWO-SAM Open Mainline
CZCWP-Tape Positioning
CZCWT-Tape Open
CZCWY-Tape Data Set Label
CZCXD-DA Output EOV
CZCXE-Mainline EOV
CZCXI-DA Input EOV
CZCXO-Tape Output EOV
CZCXS-Set DSCB
CZCXT-Tape Input EOV
CZCXX-Concatenation
CZCVA-TAM Open
CZCYG-TAM Close

CHATOS-Terminal Access Operational Status Table
used by:
CZCYM-TAM Read/Write
CZCZA-TAM Posting

CHAVPS-Virtual Program Status Word
used by:
CZCMA-GETBUF
CZCMB-GETPOOL
CZCNA-FREEBUF
CZCNB-FREEPOOL
CZCRA-BSAM Read/Write

CHAWRQ-WRITEQ
used by:
CZCTC-Terminal Task Control

QWKAR-QSAM Work Area
used by:
CZCSA-QSAM

APPENDIX B: MODULES CALLED BY ACCESS METHODS MODULES

Module	Access Methods Modules That Call this Module:
CEAAC-ADDEV	CZCYA - TAM Open CZCYG - TAM Close
CEAAD-RMDEV	CZCYA - TAM Open CZCYG - TAM Close
CEAAH-Reset	CZCMD - SETUR CZCMF - MSAM Read/Write CZCMH - MSAM Finish CZCRP - SAM Posting CZCSE - IOREQ Posting CZCZA - TAM Posting
CEAAK-SETAE	CZCRA - DA Error Retry CZCYG - TAM Close
CEAAO-I/O CALL	CZCMF - MSAM Read/Write CZCRA - BSAM Read/Write CZCRB - Control CZCRP - SAM Posting CZCSB - IOREQ CZCYM - TAM Read/Write CZCZA - TAM Posting
CEAHQ-LSCHP/ TSEND	CZCOC - MOVEPAGE CZCOD - Insert/Delete Page CZCOH - Interlock CZCOI - Release Interlock CZCOJ - Find CZCOK - Stow
CEAH3-XTRCT	CZCRH - DA Error Retry CZCYG - TAM Close
CEAH7-SETXP	CZCOC - MOVEPAGE
CEAIS-SYSER	CZCMF - MSAM Read/Write CZCMG - MSAM Posting and Error Retry CZCRP - SAM Posting CZCYG - TAM Close CZCZA - TAM Posting
CEAP4-LVPSW	CZCRA - BSAM Read/Write
CEAP7-AWAIT	CZCMI - MSAM Close CZCRA - BSAM Read/Write CZCRB - Control CZCRC - Check CZCSD - IOR Close CZCWC - SAM Close CZCWO - SAM Open Mainline CZCXE - Mainline EOV
CEAP9-TSEND	CZCYM - TAM Read/Write
CEAQ4-CKCLS	CZCMA - GETBUF CZCMC - MSAM Open CZCNB - BSAM Read/Write CZCOI - Release Interlock
CEAR0-TWAIT	CZCRC - Check
CEAA1-PGOUT	CZCOB - CLOSEVAM
CZAAB-Gate	CZCWM - Message Writer
CZABQ-WTO	CZCMD - SETUR CZCMG - MSAM Posting and Error Retry CZCMH - MSAM Finish CZCRH - DA Error Retry CZCWM - Message Writer CZCYA - TAM Open CZCYG - TAM Close CZCZA - TAM Posting
CZACP-ABEND	CZCOA - OPENVAM CZCOB - CLOSEVAM CZCOC - MOVEPAGE CZCOD - Insert/Delete Page CZCOE - REQPAGE CZCOJ - Find CZCOK - Stow CZCOM - Extend POD CZCOO - GETNUMBR CZCOP - VSAM Open CZCOR - VSAM Get CZCOS - VSAM Put CZCOT - SETL CZCOU - PUTX CZCPA - VISAM Put CZCPB - VISAM Get CZCPC - SETL CZCPE - Read/Write CZCPI - GETPAGE CZCPL - Add Directory Entry CZCPZ - VISAM Open CZCQE - Search SDST DZCYA - TAM Open CZCYG - TAM Close CZCYM - TAM Read/Write
CZAEB-FINDJFCB	CZCLA - Open Common
CZCAB-Bump	CZCXD - DA Output EOV CZCXI - DA Input EOV CZCXO - DA Input Label CZCXT - Tape Input EOV CZCXX - Concatenation CZCWD - DAOPEN CZCWT - Tape Open
CZCAP-ABEND	CZCRP - SAM Posting

Module	Access Methods Modules That Call this Module
CZCEG-GIVBKS	CZCWC - SAM Close
CZCEV-GIVBKV	CZCOB - CLOSEVAM
CZCEK-Extend	CZCOE - REQPAGE CZCXD - DA Output EOV
CZCFO-Obtain/ Retain	CZCWD - DAOPEN CZCWR - Read Format-3 DSCB CZCXD - DA Output EOV CZCXI - DA Input EOV CZCXN - DA Input Label CZCXS - Set DSCB CZCXU - DA Output Label CZCOA - OPENVAM CZCOB - CLOSEVAM
CZCGA-VMA	CZCLA - Open Common CZCLB - Close Common CZCMA - GETBUF CZCMC - MSAM Open CZCMI - MSAM Close CZCGA - VMA CZCRH - DA Error Retry CZCNB - FREEPOOL CZCOA - OPENVAM CZCOB - CLOSEVAM CZCOC - MOVEPAGE CZCOP - OPENSEQ CZCPE - Read/Write CZCPL - Add Directory Entry CZCPZ - VISAM Open CZCOM - Extend POD CZCQI - Expand RESTBL CZCSC - IOR Open CZCSD - IOR Close CZCWB - Build Common DEB CZCWC - SAM Close CZCWD - DAOPEN CZCWL - Build DA DEB CZCWO - SAM Open Mainline CZCWR - Read Format-3 DSCB CZCWT - Tape Open CZCXE - Mainline EOV CZCYA - TAM Open CZCYG - TAM Close
CZCJD-DIR	CZCMD - SETUR CZCMF - MSAM Read/Write CZCMG - MSAM Posting and Error Retry CZCMH - MSAM Finish CZCMI - MSAM Close CZCRP - SAM Posting CZCSD - IOR Close
CZCJI-INTINQ	CZCMI - MSAM Close CZCRP - SAM Posting
CZCJL-LVPRV	CZCWD - DAOPEN CZCWO - SAM Open Mainline CZCWT - Tape Open CZCWY - Tape Data Set Label CZCXN - DA Input Label CZCXU - DA Output Label CZCYA - TAM Open
CZCJS-SIR	CZCMD - SETUR CZCMG - MSAM Posting and Error Retry CZCMH - MSAM Finish CZCRH - DA Error Retry CZCRP - SAM Posting
CZCJT-QLE	CZCRH - DA Error Retry CZCRP - SAM Posting
CZCLA-Common Open	CZCMD - SETUR CZCXX - Concatenation
CZCLB-Common Close	CZCMD - SETUR CZCXX - Concatenation
CZCMA-GETBUF	CZCZA - GETBUF
CZCMC-MSAM Open	CZCLA - Open Common
CZCMF-MSAM Read/Write	CZCME-DOMSAM
CZCMH-MSAM Finish	CZCMI - MSAM Close
CZCMI-MSAM Close	CZCLB - Close Common
CZCOA-VAM Open	CZCLA - Open Common
CZCOB-VAM Close	CZCLB - Close Common
CZCOC-MOVEPAGE	CZCOA - OPENVAM CZCOB - CLOSEVAM CZCOR - VSAM Get CZCOS - VSAM Put CZCOT - SETL CZCOV - FLUSHBUF CZCPI - GETPAGE CZCQA - VISAM Close
CZCOD-Insert/ Delete Page	CZCOS - VSAM Put CZCOV - FLUSHBUF CZCPI - GETPAGE CZCPL - Add Directory Entry
CZCOE-REQPAGE	CZCOA - OPENVAM CZCOC - MOVEPAGE CZCOF - Insert
CZCOF-Insert	CZCOD - Insert/Delete Page CZCOM - Extend POD CZCOO - GETNUMBR
CZCOG-Reclaim	CZCOD - Insert/Delete Page CZCOK - Stow CZCOO - GETNUMBR
CZCOH-Interlock	CZCOA - OPENVAM CZCOC - MOVEPAGE CZCOD - Insert/Delete Page CZCOJ - Find

Module	Access Methods Modules That Call this Module
	CZCOK - Stow
	CZCOO - GETNUMBR
	CZCQE - Search SDST
CZCOI-Release Interlock	CZCOA - OPENVAM
	CZCOC - MOVEPAGE
	CZCOD - Insert/Delete Page
	CZCOJ - Find
	CZCOM - Extend POD
	CZCOO - GETNUMBR
	CZCQE - Search SDST
	CZCQQ - VAM ABEND Interlock Release
CZCOJ-Find	CZCMD - SETUR
	CZCOB - CLOSEVAM
CZCOK-Stow	CZCOJ - Find
	CZCQQ - VAM ABEND Interlock Release
CZCOL-Search	CZCOJ - Find
	CZCOK - Stow
	CZCOO - GETNUMBR
CZCOM-Extend POD	CZCOK - STOW
CZCON- Relocate Members	CZCOM - Extend POD
	CZCOO - GETNUMBR
CZCOO- GETNUMBR	CZCOC - MOVEPAGE
	CZCOD - Insert/Delete Page
CZCOP- VSAM Open	CZCOA - OPENVAM
	CZCOJ - Find
CZCOQ- VSAM Close	CZCOB - CLOSEVAM
	CZCOK - Stow
CZCOR- VSAM Get	CZCOP - VSAM Open
CZCOS- VSAM Put	CZCOP - VSAM Open
	CZCOQ - VSAM Close
	CZCOT - VSAM SETL
CZCOT-SETL	CZCOJ - Find
	CZCOP - VSAM Open
CZCOU-PUTX	CZCOP - VSAM Open
CZCOV- FLUSHBUF	CZCOQ - VSAM Close
	CZCOR - VSAM Get
	CZCOS - VSAM Put
	CZCOT - SETL
	CZCOU - PUTX
CZCPA- VISAM Put	CZCPA - VISAM Put
	CZCLB - VISAM Get
	CZCPC - SETL
	CZCPE - Read/Write; DELREC
	CZCPI - GETPAGE
	CZCQA - VISAM Close

Module	Access Methods Modules That Call this Module
CZCPB- VISAM Get	CZCPC - SETL
CZCPC-SETL	CZCOJ - Find
	CZCPE - Read/Write; DELREC
CZCPE-Read/ Write; DELREC	CZCMD - SETUR
	CZCPA - VISAM Put
	CZCPI - GETPAGE
CZCPI-GETPAGE	CZCPA - VISAM Put
	CZCPB - VISAM Get
	CZCPC - SETL
	CZCPE - Read/Write; DELREC
CZCPZ- VISAM Open	CZCOA - OPENVAM
	CZCOJ - Find
CZCQA- VISAM Close	CZCOB - CLOSEVAM
	CZCOK - Stow
CZCQE- Search SDST	CZCOA - OPENVAM
	CZCOB - CLOSEVAM
	CZCOK - Stow
	CZCQQ - VAM ABEND Interlock Release
CZCQF-JFCBVUD	CZCOB - CLOSEVAM
	CZCOE - REQPAGE
CZCQI - Expand RESTBL	CZCOA - OPENVAM
	CZCOC - MOVEPAGE
	CZCOE - REQPAGE
	CZCOJ - Find
CZCRA-SAM Read/Write	CZCRC-Check
	CZCSA-QSAM
	CZCWY-Tape Data Set Label
	CZCXS-Set DSCB
CZCRB-Control	CZCRC - Check
	CZCSA - QSAM
	CZCWY - Tape Data Set Label
	CZCXS - Set DSCB
CZCRC-Check	CZCYA - TAM Open
	CZCYG - TAM Close
	CZCSA - QSAM
CZCRG- Backspace	CZCSA - QSAM
CZCRH-DA Error Retry	CZCRP - SAM Posting
CZCRM-Point	CZCWL - Build DA DEB
	CZCSA - QSAM
CZCRN-Note	CZCSA - QSAM
CZCRQ-FINDR	CZCRG - Backspace
CZCRR-RELFUL	CZCRM - Point
CZCRS-FULREL	CZCRN - Note
	CZCXS - Set DSCB

Module	Access Methods Modules That Call this Module
CZCRX-VMER	CZCMG - MSAM Posting and Error Retry CZCRP - SAM Posting CZCZA - TAM Posting
CZCRY-VMSDR	CZCMG - MSAM Posting and Error Retry CZCRP - SAM Posting CZCZA - TAM Posting
CZCSA-QSAM FEOV	CZCLD - Force End of Volume
CZCSC-IOR Open	CZCLA - Open Common
CZCSD-IOR Close	CZCLB - Close Common CZCXI - DA Input EOV
CZCTJ-Prompt	CZCWY - Tape Data Set Label CZCXI - DA Input EOV
CZCWB-Build Common DEB	CZCWL - Build DA DEB CZCWO - SAM Open Mainline CZCWT - Tape Open CZCXE - Mainline EOV CZCXO - Tape Output EOV CZCXT - Tape Input EOV
CZCWC-SAM Close	CZCLB - Close Common
CZCWD-DAOPEN	CZCWO - SAM Open Mainline
CZCWL-Build DA DEB	CZCWD - DAOPEN CZCXD - DA Output EOV CZCXI - DA Input EOV
CZCWM-Message Writer	CZCWP - Tape Positioning CZCXD - DA Output EOV CZCXN - DA Input Label CZCXX - Concatenation
CZCWO-SAM Open	CZCLA - Open Common
CZCWP-Tape Positioning	CZCWC - SAM Close
CZCWR-Read Format-3 DSCB	CZCWD - DAOPEN CZCWX - DA Output EOV
CZCWT-Open Tape	CZCWO - SAM Open Mainline

Module	Access Methods Modules That Call this Module
CZCWV-Volume Sequence Convert	CZCWC - SAM Close CZCWO - SAM Open Mainline CZCWY - Tape Data Set Label CZCXD - DA Output EOV CZCXE - Mainline EOV CZCXI - DA Input EOV CZCXO - Tape Output EOV CZCXT - Tape Input EOV
CZCXD-DA Output EOV	CZCXE - Mainline EOV
CZCXE-SAM Mainline EOV	CZCLD - Force End of Volume CZCRC - Check CZCWC - SAM Close
CZCXI-DA Input EOV	CZCXE - Mainline EOV
CZCXN-DA Input Label	CZCWD - DAOPEN
CZCXO-TAPE Output EOV	CZCXE - Mainline EOV
CZCXS-Set DSCB	CZCWC - SAM Close CZCXD - DA Output EOV CZCXI - DA Input EOV
CZCXT-Tape Input EOV	CZCXE - Mainline EOV
CZCXU-DA Output Label	CZCWD - DAOPEN CZCXD - DA Output EOV
CZCXX-Concatenation	CZCXE - Mainline EOV CZCXI - DA Input EOV CZCXT - Tape Input EOV
CZCYA-TAM Open	CZCLA - Open Common
CZCYG-TAM Close	CZCLB - Close Common
CZCYM-TAM Write	CZCYA - TAM Open CZCYG - TAM Close CZCZA - TAM Posting
SVC-REDTIM	CZCMD - SETUR
SYSKA1-TIME	CZCWY - Tape Data Set Label

This appendix provides an alphabetic listing of the various modules that are used in the Access Methods. Also provided are the CSECT, PSECT, entry points and chart ID of each module.

Module ID	Module Name	CSECT	PSECT	Entry Points	Chart ID
CZCEI	VMIER	CZCEIC	CZCEIP	CZCEI1	JB
CZCEY	DUPOPEN	CZCEYC	CZCEYP	CZCEY1	MB
CZCEZ	DUPCLOSE	CZCEZC	CZCEZP	CZCEZ1	MF
CZCFT	DELVAM	CZCFTC	CZCFTP	CZCFT1	KF
CZCLA	OPEN COMMON	CZCLAC	CZCLAB	CZCLAO	AA
CZCLB	CLOSE COMMON	CZCLBB	CZCLBP	CZCLBC	DA
CZCLD	FORCED END OF VOLUME	CZCLDC	CZCLDB	CZCLDF	FA
CZCMA	GET A BUFFER	CZCMAC	CZCMAB	CZCMAG SYSMAG CZCMAB	HB
CZCMB	GET A BUFFER POOL	CZCMBB	CZCMBP	CZCMBG SYSMBG CZCMBP	HA
CZCMC	MSAM OPEN	CZCMCC	CZCMCP	CZCMC1	AI
CZCMD	SET UNIT RECORD	CZCMDC	CZCMDP	CZCMD1 CZCMD2 CZCMD3	AJ
CZCME	DOMSAM	CZCMEC		CZCME1 CZCME2	BB
CZCMF	MSAM READ/WRITE	CZCMFC	CZCMFP	CZCMF1	BC
CZCMG	MSAM POSTING AND ERROR RETRY	CZCMGC	CZCMGP	CZCMG1 CZCMG2	CC
CZCMH	MSAM FINISH	CZCMHC	CZCMHP	CZCMH1 CZCMH2 CZCMH3	DC
CZCMI	MSAM CLOSE	CZCMIC	CZCMIP	CZCMI1	DD
CZCNA	FREE A BUFFER	CZCMBB	CZCMBP	CZCNAF SYSNAF CZCNAP	HC
CZCNB	FREE A BUFFER POOL	CZCNBA	CZCNBB	CZCNBC SYSNBC CZCNBB	HD
CZCOA	OPEN VAM	CZCOAC	CZCOAP	CZCOA1	MA
CZCOB	CLOSE VAM	CZCOBC	CZCOBP	CZCOB1	ME
CZCOC	MOVEPAGE	CACOCC	CZCOCP	CZCOC1	JA

Module ID	Module Name	CSECT	PSECT	Entry Points	Chart ID
CZCOD	INSERT PAGE DELETE PAGE	CZCODC	CZCODP	CZCOD1 CZCOD2	KA
CZCOE	REQUEST PAGE	CZCOEC	CZCOEP	CZCOE1	KD
CZCOF	INSERT	CZCOFC	CZCOFP	CZCOF1	KB
CZCOG	RECLAIM	CZCOCC	CZCOGP	CZCOG1	KE
CZCOH	INTERLOCK	CZCOHC	CZCOHP	CZCOH1	LA
CZCOI	RELEASE INTERLOCK	CZCOIC	CZCOIP	CZCOI1	LB
CZCOJ	FIND	CZCOJC	CZCOJP	CZCOJ1	PA
CZCOK	STOW	CZCOKC	CZCOKP	CZCOK1	PB
CZCOL	SEARCH	CZCOLC	None	CZCOL1 CZCOL2	PC
CZCOM	EXTEND POD	CZCOMC	CZCOMP	CZCOM1	PD
CZCON	RELOCATE MEMBERS	CZCONC	None	CZCON1	PE
CZCOO	GETPAGE NUMBER	CZCOOC	CZCOOP	CZCOO1	PF
CZCOP	OPEN SEQUENTIAL	CZCOPC	CZCOPP	CZCOP1	MC
CZCOQ	CLOSE SEQUENTIAL	CZCOQC	CZCOQP	CZCOQ1	MG
CZCOR	VSAM GET	CZCORC	None	CZCOR1	NA
CZCOS	VSAM PUT	CZCOSC	None	CZCOS1 CZCOS2	NB
CZCOT	SET LOCATION	CZCOTC	None	CZCOT1	NC
CZCOU	PUT EXCHANGE	CZCOUC	None	CZCOU1	ND
CZCOV	FLUSH BUFFER(S)	CZCOVC	CZCOVP	CZCOV1	NE
CZCPA	VISAM PUT	CZCPAC	None	CZCPA1 CZCPA2	OA
CZCPB	VISAM GET	CZCPBC	None	CZCPB1	OB
CZCPC	SET LOCATION	CZCPCC	None	CZCPC1 CZCPC2	OC
CZCPE	READ/WRITE DELETE-RECORD	CZCPEC	CZCPEP	CZCPE1 CZCPH1	OD
CZCPI	GETPAGE END SEQUENTIAL RELEASE EXCLUSIVE	CZCPIC	CZCPIP	CZCPI1, CZCPI2, CZCPI3, CZCPD1, CZCPG1	OE
CZCPL	ADD DIRECTORY ENTRY	CZCPLC	CZCPLP	CZCPL1	OF
CZCPZ	VISAM OPEN	CZCPZC	CZCPZP	CZCPZ1	MD
CZCQA	VISAM CLOSE	CZCQAC	CZCQAP	CZCQA1	MH
CZCQE	SEARCH SDST	CZCQEC	CZCQEP	CZCQE1	JD

Module ID	Module Name	CSECT	PSECT	Entry Points	Chart ID
CZCQI	EXPAND RESTBL	CZCQEC	CZCQEP	CZCQE1	KC
CZCQK	VDMEP	CZCQKC	CZCQKP	CZCQK1 CZCQK2 CZCQK3	JC
CZCQQ	VAM ABEND INTERLOCK RELEASE	CZCQEC	CZCQEP	CZCQQ1	MI
CZCRA	BSAM READ/WRITE	CZCRAC	CZCRAP	CZCRES, CZCRAS, CZCRDS	BA
CZCRB	TAPE CONTROL	CZCRBC	CZCRBP	CZCRBS	GD
CZCRC	CHECK I/O	CZCRCC	CZCRCP	CZCRCS	CF
CZCRG	BACKSPACE	CZCRGC	CZCRGP	CZCRGA	GC
CZCRH	DA ERROR RETRY	CZCRHC	CZCRHP	CZCRH1	CB
CZCRM	LOGICALLY REPOSITION TAPE OR DA DATA SET	CZCRMC	CZCRMP	CZCRMA	GB
CZCRN	NOTE ID OF LAST RECORD READ OR WRITTEN	CZCRNS	CZCRNP	CZCRNA SYSRNA	GA
CZCRP	SAM POSTING AND ERROR RETRY	CZCRPC	CZCRPR	CZCRP1 CZCRP2	CA
CZCRQ	FIND RECORDS PER TRACK			CZCRQA	ID
CZCRR	RELFUL			CZCRRA SYSRRA	IE
CZCRS	FULREL			CZCRSA SYSRSA	IF
CZCSA	GET (The first GET following a SETL type E or B, the first GET following a FEOV, or the first GET on a data set.)			CZCSAA	QA
	PUT (Entered for the Same PUT macro instruction as GET above.)			CZCSAB	
	GET (All those not covered above.)			CZCSAG	
	PUT			CZCSAW	
	SETLB, E			CZCSAH	
	SETLR			CZCSAL	
	COMIO			CZCSAM	
	SETLP			CZCSAQ	
	SETL			CZCSAS	
	SAM CLOSE or FEOV			CZCSAV	

Module ID	Module Name	CSECT	PSECT	Entry Points	Chart ID
CZCSA	SETLC			CZCSAZ	QA
	PUTX			CZCSA2	
	TRUNC			CZCSA3	
	RELSE			CZCSA4	
	INITIO			CZCSA6	
	GET			CZCSA7	
	PUT			CZCSA8	
	TREOV			CZCSA9	
CZCSB	IOREQ	CZCSBC	CZCSBP	CZCSB1 CZCSB2	BE
CZCSC	IOR OPEN	CZCSCC	CZCSCP	CZCSC1	AL
CZCSD	IOR CLOSE	CZCSDC	CZCSDP	CZCSD1	DF
CZCSE	IOREQ POSTING	CZCSEC	CZCSER	CZCSE1	CE
CZCTC	TERMINAL TASK CONTROL	CZCTCC	CZCTCP	CZCTC1	RA
	MTT COMMAND			CZCTC1	
	FINDQ MACRO			CZCTC2	
	READQ MACRO			CZCTC3	
	WRITEQ MACRO			CZCTC4	
	CLEARQ MACRO			CZCTC5	
	FREEQ MACRO			CZCTC6	
CZCWA	ASCII TRANSLATION AND CONVERSION	CZCWAC	CZCWAP	CZCWA1	GE
CZCWB	BUILD OR MODIFY COMMON PORTION OF A DEB	CZCWBC	CZCWBP	CZCWB1	AE
CZCWC	SAM CLOSE	CZCWCC	CZCWCP	CZCWC1	DB
CZCWD	DAOPEN	CZCWDC	CZCWDP	CZCWDP	AD
CZCWL	BUILD OR EXTEND DA DEB	CZCWLC	CZCWLP	CZCWL1	AF
CZCWM	MESSAGE AND ABEND PROCESSING	CZCWMC	CZCWMP	CZCWM1	IC
CZCWO	SAM OPEN MAINLINE	CZCWOC	CZCWOP	CZCWO1	AB
CZCWP	TAPE POSITIONING	CZCWPC	CZCWPP	CZCWP1	IA
CZCWR	READ FORMAT-3 DSCBs	CZCWRC	CZCRWP	CZCRWP	AG
CZCWT	OPEN TAPE	CZCWTC	CZCWTP	CZCWT1	AC
CZCWV	VOLUME SEQUENCE CONVERT	CZCWVC	CZCWVP	CZCWV1	IB

Module ID	Module Name	CSECT	PSECT	Entry Points	Chart ID
CZCWX	TAPE VOLUME LABEL PROCESSOR	CZCWXC	CZCWXP	CZCWX1 CZCWX2 CZCWX3	EA
CZCWY	TAPE DATA SET LABEL PROCESSOR	CZCWYC	CZCWYP	CZCWY1 CZCWY2 CZCWY3 CZCWY4	EB
CZCXD	DA OUTPUT EOV	CZCXDC	CZCXDP	CZCXD1	FF
CZCXE	MAINLINE EOV	CZCXEC	CZCXEP	CZCXE1	FB
CZCXI	DA INPUT EOV	CZCXIC	CZCXIP	CZCXI1	FE
CZCXN	DA INPUT LABEL PROCESSOR	CZCXNC	CZCXNP	CZCXN1	EG
CZCXO	TAPE OUTPUT EOV	CZCXOC	CZCXOP	CZCXO1	FD
CZCXS	SET DSCB	CZCXSC	CZCXSP	CZCXS1	AH
CZCXT	TAPE INPUT EOV	CZCXTC	CZCXTP	CZCXT1	FC
CZCXU	DA OUTPUT LABEL PROCESSOR	CZCXUC	CZCXUP	CZCXU1	EH
CZCXX	CONCATENATION PROCESSOR	CZCXXC	CZCXXP	CZCXX1	FG
CZCYA	TAM OPEN	CZCYA	CZCYAP	CZCYA1	AK
CZCYG	TAM CLOSE	CZCYG	CZCYGP	CZCYG1	DE
CZCYM	TAM READ/WRITE	CZCYM	CZCYMP	CZCYM1	BD
CZCZA	TAM POSTING	CZCZA	CZCZAP	CZCZA1	CD

```
QWKAR           DSECT
QWKLEN          DC          A (QWKSZ)           QWKAR length
QWKBKL          DS          F                   Storage for backward link register
QWKFRL          DS          F                   Storage for forward link register
QWKS14          DS          F                   Storage for Register 14
QWKS15          DS          F                   Storage for Register 15
QWKSV0          DS          F                   Storage for Register 0
QWKSV1          DS          F                   Storage for Register 1
QWKSV2          DS          F                   Storage for Register 2
QWKSV3          DS          F                   Storage for Register 3
QWKSV4          DS          F                   Storage for Register 4
QWKSV5          DS          F                   Storage for Register 5
QWKSV6          DS          F                   Storage for Register 6
QWKSV7          DS          F                   Storage for Register 7
QWKSV8          DS          F                   Storage for Register 8
QWKSV9          DS          F                   Storage for Register 9
QWKS10          DS          F                   Storage for Register 10
QWKS11          DS          F                   Storage for Register 11
QWKS12          DS          F                   Storage for Register 12
QWKS13          DS          F                   Storage for Register 13
QWKRES          DS          F                   NOTE -- Reserved word number 1
QWKSTL          EQU         X'40'               SETL in progress flag
QWKCIP          EQU         X'80'               CLOSE in progress flag
DECB1           EQU         *
                DS          0F
DECB2           EQU         DECB1+DECSZ         (DECSZ is defined within CHADEC as
                                                the size of the DECB)
                DS          0F
DECB3           EQU         DECB2+DECSZ
                ORG         DECB3+DECSZ
                DS          of
QWKBKC          DS          F                   Block count of current volume
QWKRE2          DS          F                   NOTE -- Reserved word number 2
QWKGR0          DS          F                   Save for (0) between subsections
QWKGR1          DS          F                   Save for (1) between subsections
QWKGR2          DS          F                   Save for (2) between subsections
QWKGR3          DS          F                   Save for (3) between subsections
QWKGR4          DS          F                   Save for (4) between subsections
QWKGR5          DS          F                   Save for (5) between subsections
QWKGR6          DS          F                   Save for (6) between subsections
QWKGR7          DS          F                   Save for (7) between subsections
QWKWK1          DS          F                   Working area
QWKWK2          DS          F                   Register save area
QWKWK3          DS          F                   Register save area
QWKWK4          DS          F                   Register save area
QWKWK5          DS          F                   Register save area
QWKWK6          DS          F                   Register save area
QWKWK7          DS          F                   Register save area
QWKWK8          DS          F                   Working area
QWKLNK          DS          11F
QWKEND          EQU         *
QWKSZ           EQU         QWLEND--QWKLEN
```

The QWK area consists of 340 bytes; DCBQWK contains its address. The first 19 words of the work area are the QSAM general register save area. The address of this save area is furnished to any called routine in the save area parameter register. These 19 words will have the same storage protection class as the one assigned to the data set.

In the QWK work area, 36 words are used for the three DECBs used by QSAM. The first two DECBs (DECB1 and DECB2) are used for all the I/O operations. They are pointed to by the addresses in DCBDE1 and DCBDE2. All read or write operations will use the DECB pointed to by DCBDE2, and all checking operations will use the DECB pointed to by DCBDE1. The addresses in DCBDE1 and DCBDE2 are switched after each read or write so that the check will refer to the current DECB. The third DECB (DECB3) holds the appropriate copy of one of the other DECBs if a transfer to the user's SYNAD must be made. This copy is made so the user cannot erroneously alter the original DECB. The address of DECB3 is contained in DCBDE3.

APPENDIX E: DESCRIPTION OF FIELDS IN QSAM PORTION OF DCB

This discussion will not attempt to cover those fields used by QSAM which belong to the common portion of the DCB.

FIELD DISPLACEMENT FROM ZERO (in hexadecimal), QSAM USAGE

DCBRCD 90 — Address of the current logical record within the current block (1 word)

DCBEAD 94 — Address of the end of the current block (1 word)

DCBLX 98 — The complete TTRZ direct address of ZZCC magnetic tape retrieval address to be used by SETI (4 bytes)

DCBLXN 9C — Logical record count in the last block processed, used by SETL to retrieve a particular logical record within a block (2 bytes)

DCBBSV 9E — Save area for original contents of DCBBLK (2 bytes)

DCBLAD A0 — Address of the first byte of the last logical record processed (1 word)

DCBDE1 A4 — Address of DECB1 (1 word)

DCBDE2 A8 — Address of DECB2 (1 word)

DCBDE3 AC — Address of DECB3 (1 word)

DCBBF1 B0 — Address of data buffer initially assigned to DECB1 (1 word)

DCBBF2 B4 — Address of data buffer initially assigned to DECB2 (1 word)

DCBBF3 B8 — Address of data buffer used for a data set opened for RDBACK with variable format records (1 word)

DCBLRS BC — Save area for original contents of DCBLRE (2 bytes)

DCBSVL BE — Length of next logical record for variable format records in a readback data set

DCBQWK C0 — Address of the QSAM work area (1 word)

DCBQF0 C4 — Reserved for QSAM (1 byte)

DCBQF1 C5 — Field containing the following QSAM flags (1 byte):

- DCBWFL — Set on to indicate TREOV issued a write request (bit 0)
- DCBEOB — Set on to indicate an end-of-buffer condition (bit 1)
- DCBPTX — Set on by PUTX to indicate a PUTX update was issued (bit 2)
- DCBCPS — Set on to indicate to SETL that when a POINT is issued, the relative address should not be incremented by one (bit 3)
- DCBSW1 — Set on after the INITIO subsection is executed for the first time (bit 4)
- DCBLM — Set on when a locate-mode GET or PUT is issued (bit 5)
- DCBSYN — SET ON WHEN SYNAD determines that a block is to be skipped (bit 6)
- DCBAAC — Set on when SYNAD determines that an erroneous block is to be accepted (bit 7)

DCBQF2 C6 Field containing the following QSAM flags (1 byte):

DCBLSW	Set on to indicate to GET that SETL is specified in DCBMCD (bit 0)
DCBSGB	Set on to indicate that single buffering must be done (bit 1)
DCBDNN	Set on by PUT to indicate to BSAM READ/WRITE that a channel nine overflow test should be made (bit 2)
DCBDET	Set on by PUT to indicate to BSAM READ/WRITE that a channel twelve overflow test should be made (bit 3)

DCBQF3 C7 Save area for the contents of the DCBOPI field used by SETL (1 byte)

Where more than one page reference is given, the major reference is first.

IBM S/360 Access Methods Printed in U.S.A. GY28-2016-5

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)